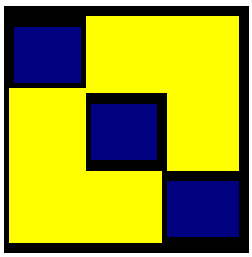
A user's guide to *MLwiN*



Jon Rasbash
William Browne
Harvey Goldstein
Min Yang
Ian Plewis
Michael Healy
Geoff Woodhouse
David Draper
Ian Langford
Toby Lewis

Version 2.1d for use with MLwiN 1.10
Centre for Multilevel Modelling
Institute of Education
University of London

A user's guide to *MLwiN*

ISBN: 085473 6123

Printed in the United Kingdom

A user's guide to *MLwiN*

by

Jon Rasbash
William Browne
Harvey Goldstein
Min Yang
Ian Plewis
Michael Healy
Geoff Woodhouse
David Draper*
Ian Langford**
Toby Lewis**

Centre for Multilevel Modelling
Institute of Education
University of London

*University of Bath
**University of East Anglia

Version 2.1d July 2002

CONTENTS

Introduction

About the Centre for Multilevel Modelling

The Centre for Multilevel Modelling consists of a team of researchers based at the Institute of Education, University of London, since 1986, funded largely by project grants from the Economic and Social Research Council (UK). The core team is as follows:

Harvey Goldstein	Professor of Statistical Methods	h.goldstein@ioe.ac.uk
Jon Rasbash	Senior research officer	j.rasbash@ioe.ac.uk
Min Yang	Senior research officer	m.yang@ioe.ac.uk
William Browne	Research officer	w.browne@ioe.ac.uk
Ian Plewis	Senior lecturer in statistics	i.plewis@ioe.ac.uk
Amy Burch	Project Administrator	a.burch@ioe.ac.uk
Lisa Brennan	Sales Administrator	l.brennan@ioe.ac.uk

In addition there are several visiting fellows and collaborators who work closely developing applications and methodology. The following are currently active:

Paul Bassett	Institute of Education	p.bassett@ioe.ac.uk
James Carpenter	London School of Hygiene	james.carpenter@lshtm.ac.uk
Diana Elbourne	Institute of Education	d.elbourne@ioe.ac.uk
Antony Fielding	University of Birmingham	a.fielding@bham.ac.uk
Michael Healy	Institute of Education	mjrhealy@compuserve.com
Dougal Hutchison	National Foundation for Educational Research	d.hutchison@nfer.ac.uk
Alastair Leyland	University of Glasgow	a.leyland@msoc.mrc.gla.ac.uk
Toby Lewis	University of East Anglia	t.lewis@uea.ac.uk
Alice McLeod	Birkbeck College	a.mcleod@psychology.bbk.ac.uk
Danny Pfefferman	University of Southampton	msdanny@vered.huji.ac.il
Nigel Rice	University of York	nr5@york.ac.uk
Vanessa Simonite	Oxford Brookes University	vsimonite@brookes.ac.uk

Centre for Multilevel Modelling
Institute of Education
20 Bedford Way
London, WC1H 0AL
England
Tel: +44 (0) 20 7612 6688
Fax: +44 (0) 20 7612 6572

Installing the *MLwiN* software

MLwiN will install under Windows 95 / 98 / 2000 / NT/ XP4. The installation procedure is as follows.

Run the file *setup.exe* from the CD-ROM. You will be guided through the installation procedure.

Once installed you simply run *MLwiN.exe,* or for example, create a shortcut menu item for it on your desktop.

An overview of *MLwiN*

MLwiN is a development from *MLn* and its precursor, ML3, which provided a system for the specification and analysis of a range of multilevel models. *MLwiN* provides a graphical user interface (GUI) for specifying and fitting a wider range of multilevel models, together with plotting, diagnostic and data manipulation facilities. The user can carry out tasks by directly manipulating GUI screen objects, for example, equations, tables and graphs.

The computing module of *MLwiN* is effectively a somewhat modified version of the DOS *MLn* program, which is driven by a series of commands and operates in the background. Users typically will set about their modelling tasks by directly manipulating the GUI screen objects. The GUI translates these user actions into *MLn* commands which are then sent to the computing module. When the computing module has completed the requested action all relevant GUI windows are notified of this and redraw themselves to reflect the updated system state. For some more complex models and tasks, for which there are currently no GUI structures available, the user must enter commands directly in the command interface window. Any commands issued by the GUI are also recorded in this window. All these commands are fully described in the *MLwiN* **Help** system (see below).

It is assumed that you have a working knowledge of Windows 95/98 or NT applications. The *MLwiN* interface shares many features common to other applications such as word processors and some statistical packages. Thus, file opening and saving is standard, as is the arranging and copying of windows to the clipboard, and using menus and dialogue boxes.

The data structure is essentially that of a spreadsheet or worksheet with columns denoting variables and rows corresponding to the lowest level units in the hierarchy. For example in the exam data set described in Chapter 2 there are 4059 rows, one for each student, and there are columns identifying the student and school and containing the values of the variables to be used in the analysis. By default the program allocates 400 columns, 150 fixed and 150 random parameters and 5 levels of nesting. The worksheet dimensions, the number of parameters and the number of levels can be allocated dynamically.

For your own data analysis, typically you will have prepared your data in rows (or records) corresponding to the cases you have observed. *MLwiN* enables such data to be read into separate columns of a new worksheet, one column for each field. Data input and output is accessed from the **File** menu.

Other columns may be used for other purposes, for example to hold frequency data or parameter estimates, and need not be of the same length. Columns are numbered and can be referred to either as C1, C17, C43 etc., or by name if they have previously been named using the NAME feature in the **Names** window. *MLwiN* also has groups whose elements are a list of columns. These are fully described in the *MLwiN* **Help** system.

As well as the columns there are also ‘boxes’ or constants, referred to as B1, B10 etc. *MLwiN* is not case-sensitive so it will be most convenient for you to type in lower case

although you may find it useful to adopt a convention of using capital letters and punctuation for annotating what you are doing.

Enhancements in Version 1.1

MLwiN Help

The basic reference for *MLwiN* is provided by an extensive Help system. This uses the standard Windows 95/NT Help conventions. Links are underlined in green and 'topics' are listed under 'books'. There is a principal Help button located on the main menu and context sensitive buttons located on individual screens. You can use the 'index' to search for a topic or alternatively if you click on the 'find' tab you can search using keywords for the topic. Navigating through the Help system is done by clicking on hypertext links, or using any of the options on the Help screen menu bars. You can also use any of the functions available under 'options' on the windows 95/NT Help toolbar, such as annotating Help pages, printing, changing font size, etc.

Compatibility with existing *MLn* software

It is possible to use *MLwiN* in just the same way as in *MLn* via the **Command interface** window. Opening this and clicking on the **Output** button allows you to enter commands and see the results in a separate window. For certain kinds of analysis this is the *only* way to proceed. *MLwiN* will read existing *MLn* worksheets, and a switch can be set when saving *MLwiN* worksheets so that they can be read by *MLn*. For details of all *MLwiN* commands see the relevant topics in the Help system. You can access these in the index by typing "command name" where 'name' is the *MLn* command name.

Macros

MLwiN contains enhanced macro functions which allow users to design their own menu interfaces for specific analyses. A special set of macros for fitting discrete response data using quasilikelihood estimation has been embedded into the **Equations window** interface so that the fitting of these models is now entirely consistent with the fitting of Normal models. A full discussion of macros is given in the *MLwiN* Help system.

Prerequisites and the structure of the user's guide

Following this introduction the first chapter provides an introduction to multilevel modelling, its aims and the formulation of a simple model. A key, innovative, feature of *MLwiN* is the **Equations window** which allows the user to specify and manipulate a model using standard statistical notation. This assumes that users of *MLwiN* will have a statistical background which encompasses a basic understanding of multiple regression analysis and the corresponding standard notation associated with that. In the next chapter we introduce multilevel modelling by developing a multilevel model building upon a simple regression model. After that there is a detailed analysis of an educational data set which introduces the key features of *MLwiN*. There follow a series of further

chapters which take users through the analysis of different kinds of data, illustrating further features of *MLwiN*, including its more advanced ones. The user's guide includes two appendices, on cross classifications and multiple membership models, which describe how to fit these models using *MLwiN* commands.

We suggest that users to take the time to work through at least the first tutorial to become familiar with the software. The **Help** system is extensive and covers all the available features with full explanations of *MLwiN* features and also help with many of the statistical procedures. Reduced versions of the tutorials are also available within the Help system.

Acknowledgements

The development of the *MLwiN* software has been the principal responsibility of Jon Rasbash, but also owes much to the efforts of a number of people outside the Centre for Multilevel Modelling.

Michael Healy developed the program NANOSTAT which was the precursor of *MLn* and hence *MLwiN* and we owe him a considerable debt for his inspiration and continuing help. William Browne wrote the basic code for the MCMC modelling with advice from David Draper. The Economic and Social Research Council (ESRC) has provided continuous support to the Centre for Multilevel Modelling at the Institute of Education since 1986, most recently as part of its Analysis of Large and Complex Datasets programme directed by Fred Smith: without their support *MLwiN* could not have happened. A number of visiting fellows have been funded by ESRC: Ian Langford, Alastair Leyland, Toby Lewis, Dick Wiggins, Dougal Hutchison, Nigel Rice and Tony Fielding; they have contributed greatly.

Many others, too numerous to mention, have played their part and we particularly would like to acknowledge the stimulation and encouragement we have received from the team at the MRC Biostatistics unit in Cambridge - the BUGS software developments have complemented our own efforts. We are also most grateful to the Joint Information Systems Committee (U.K.) for funding a project related to parallel processing procedures for multilevel modelling.

Further information about multilevel modelling

There is a Web site which contains much of interest, including new developments, and details of courses and workshops. The site is located at the Institute of Education in London. To view this go to the following address:

http://multilevel.ioe.ac.uk/

This web site also contains the latest information about *MLwiN* software, including upgrade information, maintenance downloads, and documentation.

There is an active email discussion group about multilevel modelling. You can join this by sending email to jiscmail@jiscmail.ac.uk with a single message line as follows: (Substituting your own first and last names for *firstname* and *lastname*)

Join multilevel firstname lastname

The Centre for Multilevel Modelling in London also issues a newsletter twice a year and you can join the free mailing list by sending email to a.burch@ioe.ac.uk.

Technical Support

For technical support of any kind please send an email, setting out your problem, to Amy Burch a.burch@ioe.ac.uk.

Chapter 1. Introducing multilevel models

Multilevel data structures

In the social, medical and biological sciences multilevel or hierarchically structured data are the norm and they are also found in many other areas of application. For example, school education provides a clear case of a system in which individuals are subject to the influences of grouping. Pupils or students learn in classes; classes are taught within schools; and schools may be administered within local authorities or school boards. The *units* in such a system lie at four different levels of a hierarchy. A typical multilevel model of this system would assign pupils to level 1, classes to level 2, schools to level 3 and authorities or boards to level 4. Units at one level are recognised as being grouped, or nested, within units at the next higher level.

In a household survey, the level 1 units are individual people, the level 2 units are households and the level-3 units, areas defined in different ways. Such a hierarchy is often described in terms of *clusters* of level 1 units within each level 2 unit etc. and the term *clustered population* is used.

In animal or child growth studies repeated measurements of, say, weight are taken on a sample of individuals. Although this may seem to constitute a different kind of structure from that of a survey of school students, it can be regarded as a 2-level hierarchy, with animals or children at level 2 and the set of measurement occasions for an individual constituting the level 1 units for that level 2 unit. A third level can be introduced into this structure if children are grouped into schools or young animals grouped into litters.

In trials of medical procedures, several centres may be chosen and individual patients studied within each one. Here the centres become the level 2 units and the patients the

level 1 units.

In all these cases, we can see clear hierarchical structures in the population. From the point of view of our models what matters is how this structure affects the measurements of interest. Thus, if we are measuring educational achievement, it is known that average achievement varies from one school to another. This means that students within a school will be more alike, on average, than students from different schools. Likewise, people within a household will tend to share similar attitudes etc. so that studies of, say, voting intention need to recognise this. In medicine it is known that centres differ in terms of patient care, case mix, etc. and again our analysis should recognise this.

Consequences of ignoring a multilevel structure

The point of multilevel modelling is that a statistical model explicitly should recognise a hierarchical structure where one is present: if this is not done then we need to be aware of the consequences of failing to do this.

In our first tutorial example we look at the relationship between an outcome or response variable which is the score achieved by 16 year old students in an examination and a predictor or explanatory variable which is a reading test score obtained by the same students just before they entered secondary school at the age of 11 years. The first variable is referred to as "exam score" and the second as "LRT score" where LRT is short for 'London Reading Test'. In the past it would have been necessary to decide whether to carry out this analysis at school level or at pupil level. Both of these single-level analyses are unsatisfactory, as we now show.

In a school-level or aggregate analysis, the mean exam score and the mean LRT score would first be calculated for each school. Ordinary regression would then be used to estimate a relationship between these. The main problem here is that it is far from clear how to interpret any relationship found. Any causal interpretation must include individual pupils, and information about these has been discarded. In practice it is possible to find a wide variety of models, each fitting the data equally well, but giving widely different estimates. Because of the difficulty of interpretation the results of such analyses depend on an essentially arbitrary choice of model. An empirical demonstration of this unreliability is given by Woodhouse and Goldstein (1989), who analyse examination results in English Local Education Authorities.

In a pupil-level analysis an average relationship between the scores would be estimated using data for all 4059 pupils. The variation between schools could be modelled by incorporating separate terms for each school. This procedure is inefficient, and inadequate for the purpose of generalisation. It is inefficient because it involves estimating many times more coefficients than the multilevel procedure; and because it does not treat schools as a random sample it provides no useful quantification of the variation among schools in the population more generally.

By focusing attention on the levels of hierarchy in the population, multilevel modelling enables the researcher to understand where and how effects are occurring. It provides better estimates in answer to the simple questions for which single-level analyses were once used and in addition allows more complex questions to be addressed. For example Nuttall *et al* (1989), using multilevel modelling, showed that secondary schools varied in the progress made by students from different ethnic groups: in some schools certain ethnic minority group children made more progress, in comparison with non-minority children, than in other schools.

Finally, carrying out an analysis which does not recognise the existence of clustering at all, for example a pupil level analysis with no school terms, creates serious technical problems. For example, ignored clustering will generally cause standard errors of regression coefficients to be underestimated. Consider also models of electoral behaviour. Voters are clustered within wards and wards within constituencies. If standard errors were underestimated it might be inferred, for example, that there was a real preference for one party or course of action over another when in fact that preference, estimated from the sample, could be ascribed to chance. Correct standard errors would be estimated only if variation at ward and constituency level were allowed for in the analysis. Multilevel modelling provides an efficient way of doing this. It also makes it possible to model and investigate the relative sizes and effects of ward characteristics and of constituency characteristics on electoral behaviour, as well as that of individual characteristics such as social group.

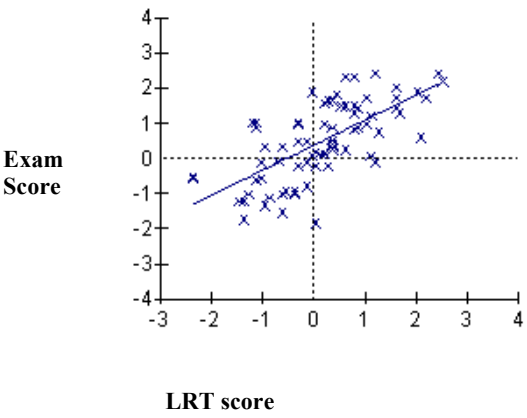
There is now a considerable literature on multilevel modelling, both theoretical and applied. The tutorial examples will enable the new user to become familiar with the basic concepts. More detailed discussions and statistical derivations can be found in the books by Bryk and Raudenbush (1992), Longford (1993) and Goldstein (1995a).

Levels of a data structure

We shall use our exam data to illustrate the fundamental principle of multilevel modelling: the existence of different levels of variation. For this purpose schools will be the groups of interest.

We start by introducing some basic terminology and to keep matters simple we restrict attention to a single response variable and one predictor. We begin by looking at the data from a single school. In Figure 1.1 the exam scores of 73 children sampled from one of the schools in our sample are plotted against the LRT scores for the same children. The relationship is summarised by the prediction or regression line.

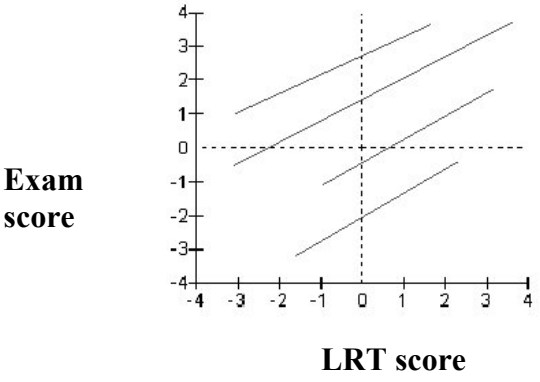
Figure 1.1 Level 1 variation



Each of the graph points represents a pair of values for an individual student. The degree of scatter about the regression line represents random variation at the student level about the average relationship between exam and LRT scores at this school. This is the level 1 variation because it concerns the variation between level 1 units, that is students.

We can think of the regression line as a school-level summary of this relationship which may itself vary from school to school. In Figure 1.2 we have drawn the regression lines for four typical schools, constraining the lines to be parallel.

Figure 1.2 Level 2 variation in school summary lines

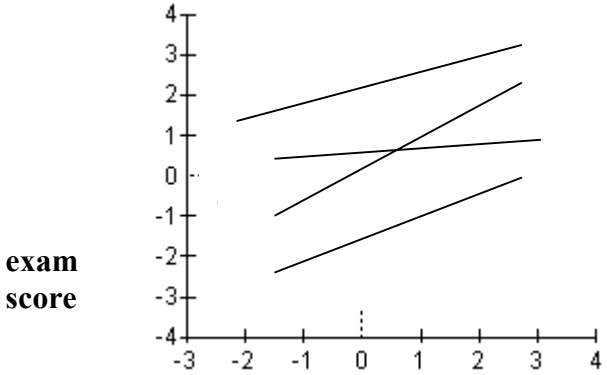


The lines in Figure 1.2 have different *intercepts*. The variation between these intercepts is called level 2 variation because in this example the schools are level 2 units. The schools are thought of as a random sample from a large underlying population of schools and 'school' is referred to as a *random* classification. The individual schools, like the individual pupils, are not of primary interest. Our interest is rather to make inferences about the variation among all schools in the population, using the sampled schools as a means to that end.

If we regard the lines in Figure 1.2 as giving the prediction of the exam score for a given LRT score, then it is clear that the differences between schools is constant across the range of LRT scores. This kind of variation is known as *simple* level 2 variation.

If we allow the slopes of the lines to vary as in Figure 1.3, then the differences between the schools depend on the students' LRT scores. This is an example of *complex* level 2 variation.

Figure 1.3 Complex level 2 variation



Once again, the main focus of a multilevel analysis is not on the individual schools in the sample, but on estimating the pattern of variation in the underlying population of schools. Once this is done it becomes possible to attempt to explain the pattern in terms of general characteristics of schools, by incorporating further variables into the model. We can also obtain 'posterior' estimates of intercepts and slopes for each school and the procedures for this will be illustrated in the next chapter.

Before we set up a simple model for the examination data we briefly review the basic statistical theory of multilevel modelling. This section can be skipped by those familiar with the statistical background.

An introductory description of multilevel modelling

Figure 1.1 provided an illustration of level 1 variation for a single school, together with a regression line representing a summary relationship between the exam and LRT scores for the pupils in this school. The technique of 'ordinary least squares', or OLS, to produce this relationship is well known and provided by many computer packages, including *MLwiN*. Our interest, however, is to use the variation between all the schools of the sample in order to make inferences about the variation in the underlying population.

We use the regression line in order to revise some standard algebraic notation. The ordinary regression relationship for a single school may be expressed as:

$$y_i = a + bx_i + e_i \tag{1.1}$$

where subscript *i* takes values from 1 to the number of pupils in the school. In our example y_i and x_i are respectively the exam and LRT scores for the *i*-th pupil. The regression relation can also be expressed as:

$$\hat{y}_i = a + bx_i \tag{1.2}$$

where $\hat{y}_i$ is the exam score predicted for the *i*-th pupil by this particular summary relationship for the school? The intercept, *a*, is where the regression line meets the vertical axis and *b* is its slope. The expression $a + bx_i$ is known as the *fixed part* of the model.

In equation (1.1), e_i is the departure of the *i*-th pupil's actual exam score from the predicted score. It is commonly referred to either as an *error* or as a *residual.* In this volume we shall use the latter term. This residual is that part of the score y_i which is not predicted by the fixed part regression relationship in equation (1.2). With only one school, the level 1 variation is just the variance of these e_i.

Turning now to the multilevel case of several schools which are regarded as a random sample of schools from a population of schools, we assume a regression relation for each school

$$\hat{y}_{i1} = a_1 + bx_{i1}$$
$$\hat{y}_{i2} = a_2 + bx_{i2}$$
……

where the slopes are parallel. More succinctly, we can write this as

$$\hat{y}_{ij} = a_j + bx_{ij} \tag{1.3}$$

The subscript *j* takes values from 1 to the number of schools in the sample.

We can write the full model as

$$y_{ij} = a_j + bx_{ij} + e_{ij} \tag{1.4}$$

In general, wherever an item has two subscripts *ij,* it varies from pupil to pupil within a school. Where an item has a *j* subscript only it varies across schools but has the same

value for all the pupils within a school. And where an item has neither subscript it is constant across all pupils and schools.

In a multilevel analysis the level 2 groups, in this case schools, are treated as a random sample from a population of schools. We therefore re-express equation (1.3) as

$$a_j = a + u_j \qquad (1.5)$$
$$\hat{y}_{ij} = a + bx_{ij} + u_j$$

Where *a*, with no subscripts, is a constant and u_j, the departure of the *j*-th school's intercept from the overall value, is a level 2 residual which is the same for all pupils in school *j*.

The model for actual scores can now be expressed as:

$$y_{ij} = a + bx_{ij} + u_j + e_{ij} \qquad (1.6)$$

In this equation, both u_j and e_{ij} are random quantities, whose means are equal to zero; they form the *random part* of the model (1.6). We assume that, being at different levels, these variables are uncorrelated and we further make the standard assumption that they follow a Normal distribution so that it is sufficient to estimate their variances, σ_u^2 and σ_e^2 respectively. The quantities a and b, the mean intercept and slope, are fixed and will also need to be estimated.

It is the existence of the two random variables u_j and e_{ij} in equation (1.6) which marks it out as a multilevel model. The variances σ_u^2 and σ_e^2 are referred to as *random parameters* of the model. The quantities *a* and *b* are known as *fixed parameters*.

A multilevel model of this simple type, where the only random parameters are the intercept variances at each level, is known as a variance components model.

In order to specify more general models we need to adapt the notation of (1.6). First, we introduce a special explanatory variable x_0, which takes the value 1 for all students. This allows every term on the right hand side of (1.6) to be associated with an explanatory variable. Secondly, we use β_0, β_1 etc. for the fixed parameters, the subscripts 0, 1 etc. matching the subscripts of the explanatory variables to which they are attached. Similarly, we incorporate a subscript 0 into the random variables and write

$$y_{ij} = \beta_0 x_0 + \beta_1 x_{1ij} + u_{0j} x_0 + e_{0ij} x_0 \tag{1.7}$$

Finally, we collect the coefficients together and write

$$\begin{aligned} y_{ij} &= \beta_{0ij} x_0 + \beta_1 x_{1ij} \\ \beta_{0ij} &= \beta_0 + u_{0j} + e_{0ij} \end{aligned} \tag{1.8}$$

Thus we have specified the random variation in y in terms of random coefficients of explanatory variables. In the present case the coefficient of x_0 is random at both level 1 and level 2. The zero subscripts on the level 1 and level 2 random variables indicate that these are attached to x_0.

For the standard model we assume that the response variable is Normally distributed and this is usually written in standard notation as follows

$$y \sim N(XB, \Omega) \tag{1.9}$$

where XB is the fixed part of the model and in the present case is a column vector beginning:

$$\begin{pmatrix} \beta_0 x_{011} + \beta_1 x_{111} \\ \beta_0 x_{021} + \beta_1 x_{121} \\ \cdots\cdots\cdots \end{pmatrix}$$

The symbol Ω represents the variances and covariances of the random terms over all the levels of the data. In the present case it denotes just the variances at level 1 and level 2.

Equations (1.8) and (1.9) form a complete specification for our model and *MLwiN* uses this notation to specify the models which are described in the following chapters.

PART 1. FITTING TWO LEVEL MODELS

Chapter 2: Random Intercept and Random Slope Models

This chapter is a tutorial which will take you through the basic procedures for specifying a multilevel model in *MLwiN*, estimating parameters, making inferences, and plotting results. It provides both an introduction to the software and a practical introduction to multilevel modelling.

As we have seen, multilevel models are useful in a very wide range of applications. For illustration here, we use an educational data set for which an *MLwiN* worksheet has already been prepared. Usually, at the beginning of an analysis, you will have to create such a worksheet yourself either by entering the data directly or by reading a file or files prepared elsewhere. Facilities for doing this are described at the end of this chapter. The data in the worksheet we use have been selected from a very much larger data set, of examination results from six inner London Education Authorities (school boards). A key aim of the original analysis was to establish whether some schools were more 'effective' than others in promoting students' learning and development, taking account of variations in the characteristics of students when they started Secondary school. The analysis then looked for factors associated with any school differences found. Thus the focus was on an analysis of factors associated with examination performance after adjusting for student intake achievements. As you explore *MLwiN* using the simplified data set you will also be imitating, in a simplified way, the procedures of the original analysis. For a full account of that analysis see Goldstein *et al.* (1993).

Opening the worksheet and looking at the data

When you start *MLwiN* the main window appears. Immediately below the *MLwiN* title bar are the *menu bar* and below it the *tool bar* as shown:

These menus are fully described in the online Help system. This may be accessed either by clicking the **Help** button on the menu bar shown above or (for context-sensitive Help) by clicking the **Help** button displayed in the window you are currently working with. You should use this system freely.

The buttons on the tool bar relate to model estimation and control, and we shall describe these in detail later. Below the tool bar is a blank workspace into which you will open windows using the **Window** menu. These windows form the rest of the 'graphical user interface' which you use to specify tasks to *MLwiN*. Below the workspace is the *status bar*, which monitors the progress of the iterative estimation procedure. Open the tutorial worksheet as follows:

Select **File** menu

Select **Open worksheet**

Select **tutorial.ws**

Click **Open**

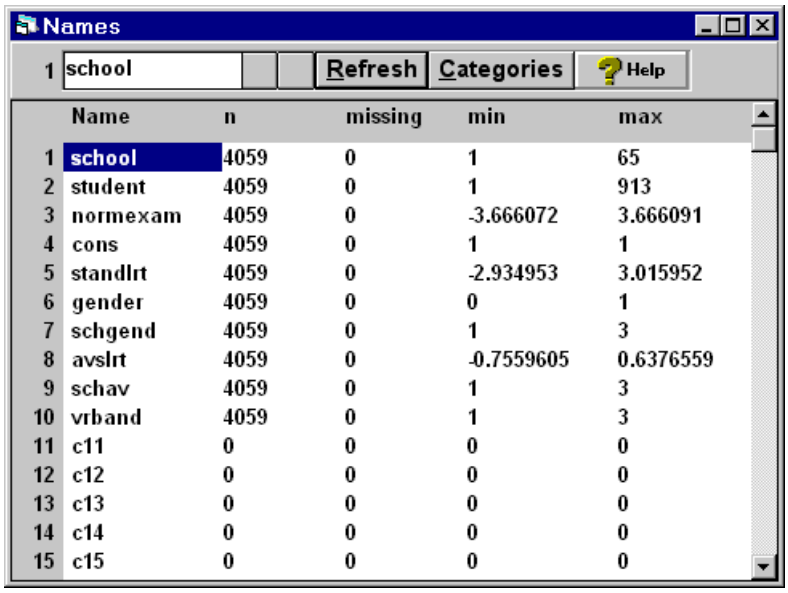
When this operation is complete the filename will appear in the title bar of the main window and the status bar will be initialised.

The *MLwiN* worksheet holds the data and other information in a series of *columns*. These are initially named c1, c2, …, but the columns can (and should) be given meaningful names to show what their contents relate to. This has already been done in the **Tutorial** worksheet that you have loaded. You can view a summary of the data in the worksheet as follows:

Select **data manipulation**

Select **Names**

This opens the following window:

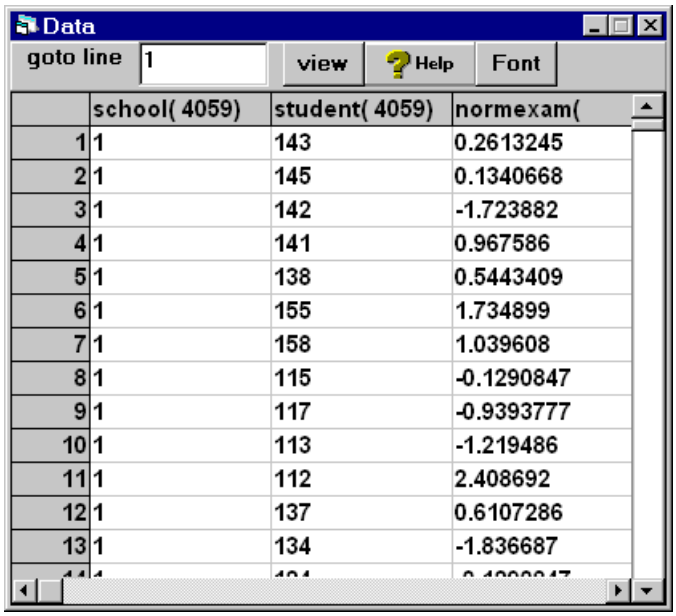


	Name	n	missing	min	max
1	school	4059	0	1	65
2	student	4059	0	1	913
3	normexam	4059	0	-3.666072	3.666091
4	cons	4059	0	1	1
5	standlrt	4059	0	-2.934953	3.015952
6	gender	4059	0	0	1
7	schgend	4059	0	1	3
8	avslrt	4059	0	-0.7559605	0.6376559
9	schav	4059	0	1	3
10	vrband	4059	0	1	3
11	c11	0	0	0	0
12	c12	0	0	0	0
13	c13	0	0	0	0
14	c14	0	0	0	0
15	c15	0	0	0	0

Each line in the body of the window summarises a column of data. In the present case only the first 10 of the 400 columns of the worksheet contain data. Each column contains 4059 items, one item for each student represented in the data set. There are no missing values, and the minimum and maximum value in each column are shown. Note the Help button on the tool bar. The remaining items on the tool bar of this window are for attaching a name to a column. We shall use these later.

You can view individual items in the data using the Data window as follows:

Select **Data manipulation** menu Select **View or edit data**



	school(4059)	student(4059)	normexam(
1	1	143	0.2613245
2	1	145	0.1340668
3	1	142	-1.723882
4	1	141	0.967586
5	1	138	0.5443409
6	1	155	1.734899
7	1	158	1.039608
8	1	115	-0.1290847
9	1	117	-0.9393777
10	1	113	-1.219486
11	1	112	2.408692
12	1	137	0.6107286
13	1	134	-1.836687

When this window is first opened it always shows the first three columns in the worksheet. The exact number of items shown depends on the space available on your screen.

You can view any selection of columns, spreadsheet fashion, as follows:

Click the **View** button

Select columns to view

Click **OK**

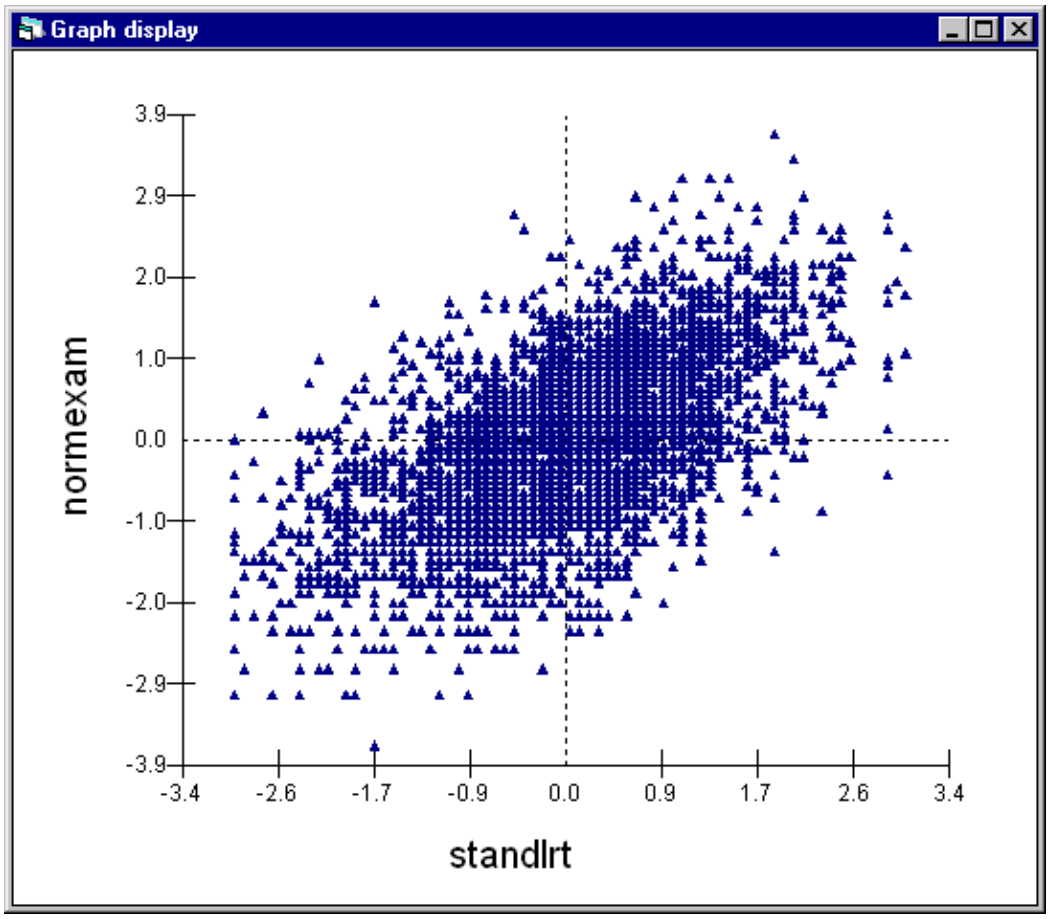
You can select a block of adjacent columns either by pointing and dragging or by selecting the column at one end of the block and holding down 'Shift' while you select the column at the other end. You can add to an existing selection by holding down 'Ctrl' while you select new columns or blocks.

The **Font** button, which is present in several of the *MLwiN* windows, can be used to make the characters *in that window* larger or smaller. This can be useful when the space available for the windows is not too large.

The **school** and **student** columns contain identifiers; **normexam** is the exam score obtained by each student at age 16, Normalised to have approximately a standard Normal distribution, **cons** is a column of 1's, and **standlrt** is the score for each student

at age 11 on the London Reading Test, standardised using *z*-scores. **Normexam** is going to be the y-variable and **cons** and **standlrt** the x-variables in our initial analysis. The other data columns will be used in later sections of the manual. Use the scroll bars of the **Data** window to move horizontally and vertically through the data, and move or resize the window if you wish. You can go straight to line 1035, for example, by typing 1035 in the **goto line** box, and you can highlight a particular cell by pointing and clicking. This provides a means to edit data: see the Help system for more details.

Having viewed your data you will typically wish to tabulate and plot selected variables, and derive other summary statistics, before proceeding to multilevel modelling. Tabulation and other basic statistical operations are available on the **basic statistics** menu. These operations are described in the help system. In our first model we shall be looking at the relationship between the outcome attainment measure **normexam** and the intake ability measure **standlrt** and at how this relationship varies across schools. The scatter plot of **normexam** against **standlrt** for the whole of the data looks like this:



The plot shows, as might be expected, a positive correlation with pupils with higher intake scores tending to have higher outcome scores. Our modelling will attempt to partition the overall variability shown here into a part which is attributable to schools and a part which is attributable to students. We will demonstrate later in the chapter how to produce such graphs in *MLwiN* but first we focus on setting up a basic model.

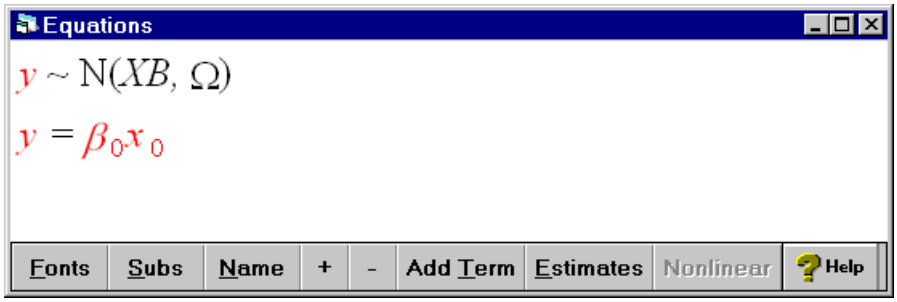
You can now proceed straight away to the next section of this chapter, or stop at this point and close *MLwiN*. No data have been changed and you can continue with the next section after re-opening the worksheet **Tutorial.ws**. Each of the remaining sections in this chapter is self-contained [but they must be read in the right order!], and you are invited to save the current worksheet (using a different name) where necessary to preserve continuity.

Setting up a variance components multilevel model

We now go through the process of specifying a two-level variance components model for the examination data. First, close any open windows in the workspace. Then:

Select **Model** menu Select **Equations**

The following window appears:



This window shows the nucleus of a model, which you elaborate in stages to specify the one you want. The tool bar for this window is at the bottom, and we shall describe these buttons shortly.

The first line in the main body of the window specifies the default distributional assumption: the response vector has a mean specified in matrix notation by the *fixed part* XB, and a random part consisting of a set of random variables described by the *covariance matrix* Ω. This covariance matrix Ω incorporates the separate covariance matrices of the random coefficient at each level. We shall see below how it is specified. Note that y and x_0 are shown in red. This indicates that they have not yet been defined.

To define the response variable we have to specify its name and also that there are two levels. The lowest level, level 1, represents the variability between students at the same

school; the next higher level, level 2, represents the variability between different schools. To do all this

Click *y* (either of the *y* symbols shown will do)

The **Y variable** dialogue box appears, with two drop-down lists: one labelled **y**, the other labelled **N levels**.

In the **y** list, select **normexam** In the **N levels** list, select **2-ij**

By convention, the suffix *i* is used by *MLwiN* for level 1 and *j* for level 2, but suffixes can be changed as we will show later.

This reveals two further drop-down lists, *level 2 (j)* and *level 1 (i).*

In the *level 2 (j)* list, select **school** In the *level 1 (i)* list, select **student** Click **done**

In the **Equations** window the red *y* has changed to y_{ij} in black indicating that the response and the number of levels have been defined.

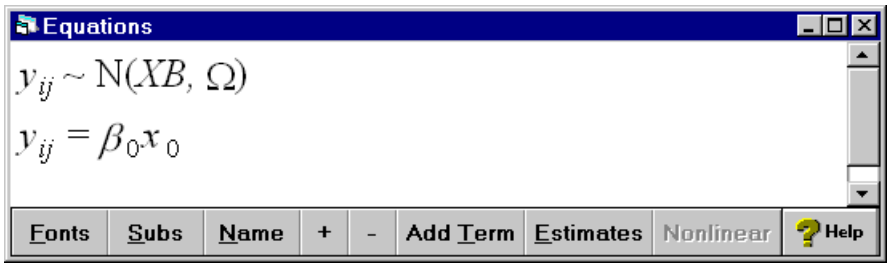
Now we must define the explanatory variables.

Click x_0 In the drop-down list, select **cons**

Note that the **fixed parameter** box is checked: by default, each explanatory variable is assumed to have a fixed parameter. We have just identified the explanatory variable x_0 with a column of 1's. This vector of 1's explicitly models the intercept. Other software packages may do this for you automatically, however, in the interests of greater flexibility, *MLwiN* does not.

Click **Done**

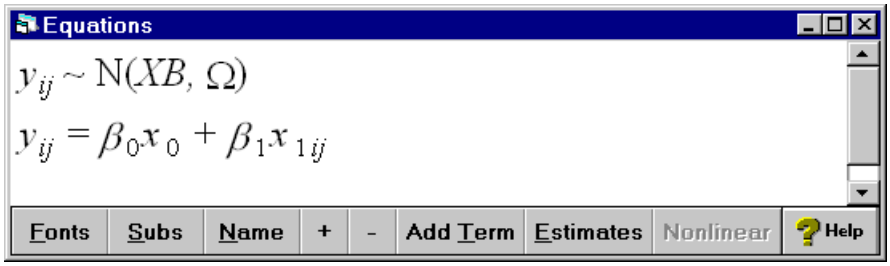
The **Equations** window now looks like this:



We are gradually building equation (1.8) which assumes the simple level 2 variation shown in figure 1.2. We have specified the fixed parameter associated with the intercept, and now require another explanatory variable.

Click the **AddTerm** button on the tool bar Click x_1 Select **standlrt** Click **Done**

The **Equations** window looks like this –



This completes the specification of the fixed part of the model. Note that x_0 has no other subscript but that x_1 has collected subscripts ij. *MLwiN* detects that **cons** is constant over the whole data set, whereas the values of **standlrt** change at both level 1 and level 2.

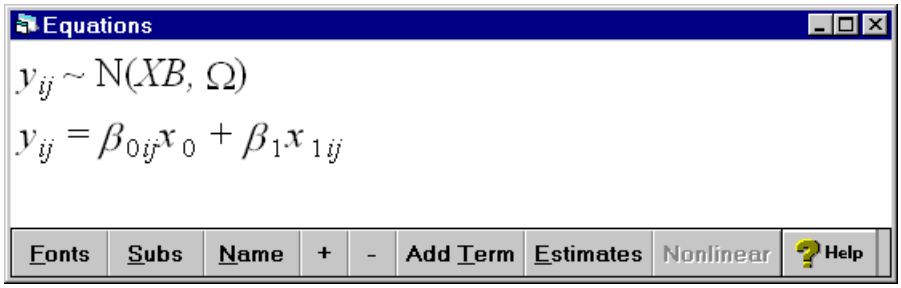
To define the random part.

Click β_0 (or x_0)

This redisplays the dialogue box for x_0, seen earlier. We wish to specify that the coefficient of x_0 is random at both school and student levels.

Check the box labelled **j(SCHOOL)** Check the box labelled **i(STUDENT)** Click **Done**

This produces

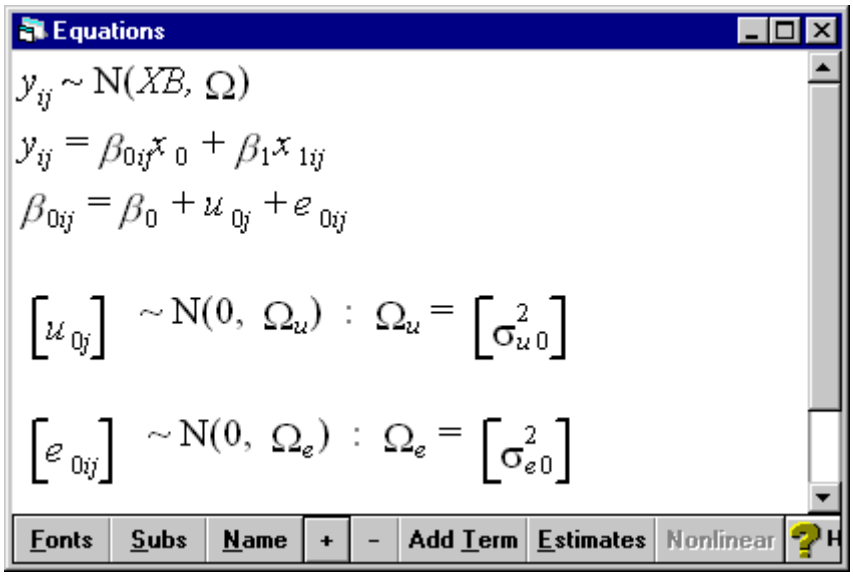


We have now defined the model. To see the composition of β_{0ij},

Click the + button on the tool bar

You should now see the model as defined in equation (1.8).

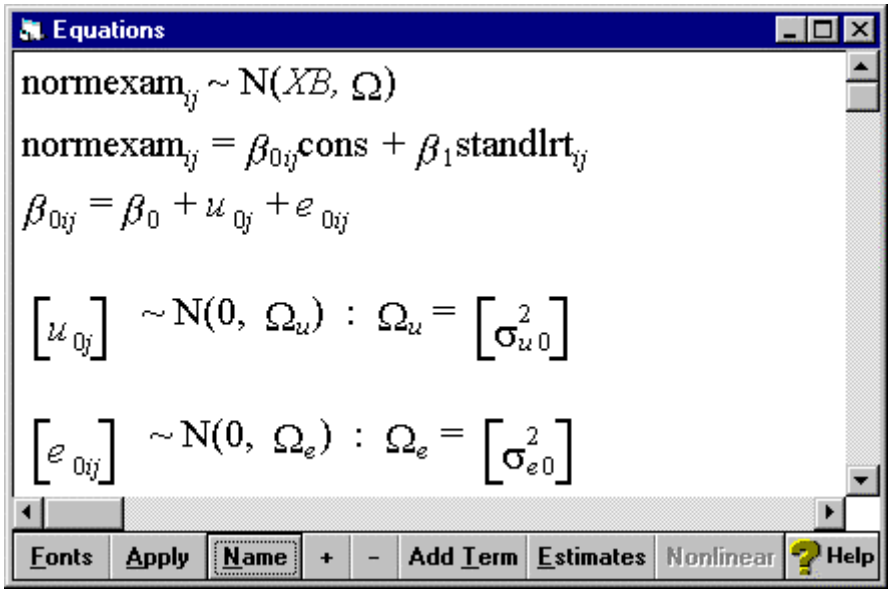
The + and – buttons control how much detail of the model is displayed. Click + a second time to reveal:



You may need to resize the window by dragging the lower border in order to see all the details, or alternatively change the font size.

To replace y, x_0 and x_1 by their variable names,

Click the **Name** button



The **Name** button is a 'toggle': clicking again brings back the x's and y's.

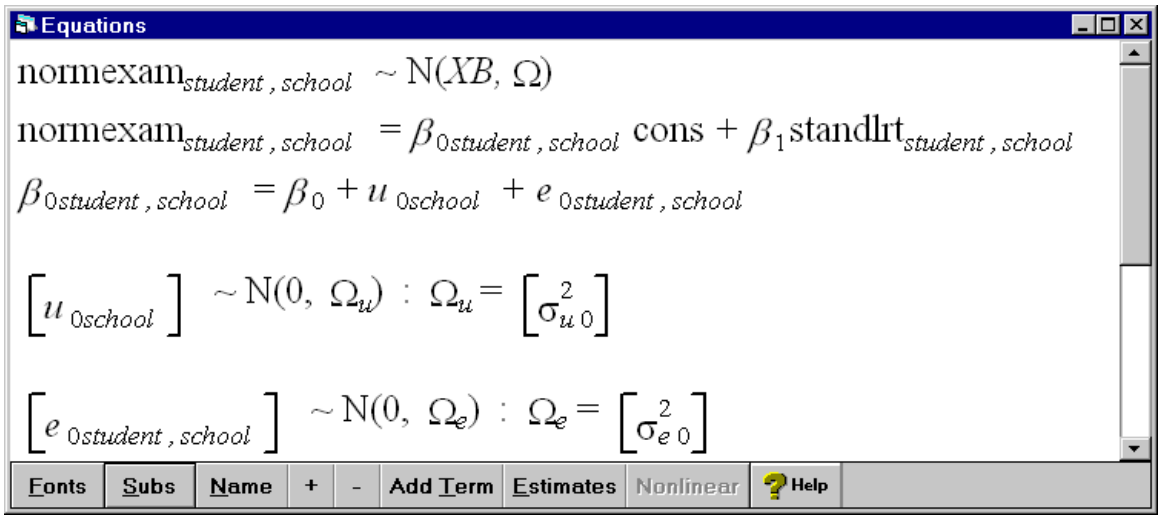
In summary, the model that we have specified relates **normexam** to **standlrt.** The regression coefficients for the intercept and the slope of **standlrt** are (β_0, β_1). These

coefficients define the average line across all students in all schools. The model is made multilevel by allowing each school's summary line to depart (be raised or lowered) from the average line by an amount u_{0j}. The *i*'th student in the *j*'th school departs from its school's summary line by an amount e_{0ij}. The information conveyed on the last two lines of the display is that the school level random departures u_{0j} are distributed Normally with mean 0 and variance σ^2_{u0} and the student level random departures e_{0ij} are distributed Normally with mean 0 and variance σ^2_{e0} (the Ω's can be ignored for the time being). The u_{0j} (one for each school) are called the level 2 or school level *residuals*; the e_{0ij} (one for each student) are the level 1 or student level residuals.

Just as we can toggle between *x*'s and actual variable names, so we can show actual variable names as subscripts. To do this

Click the **Subscripts** button

Which produces:



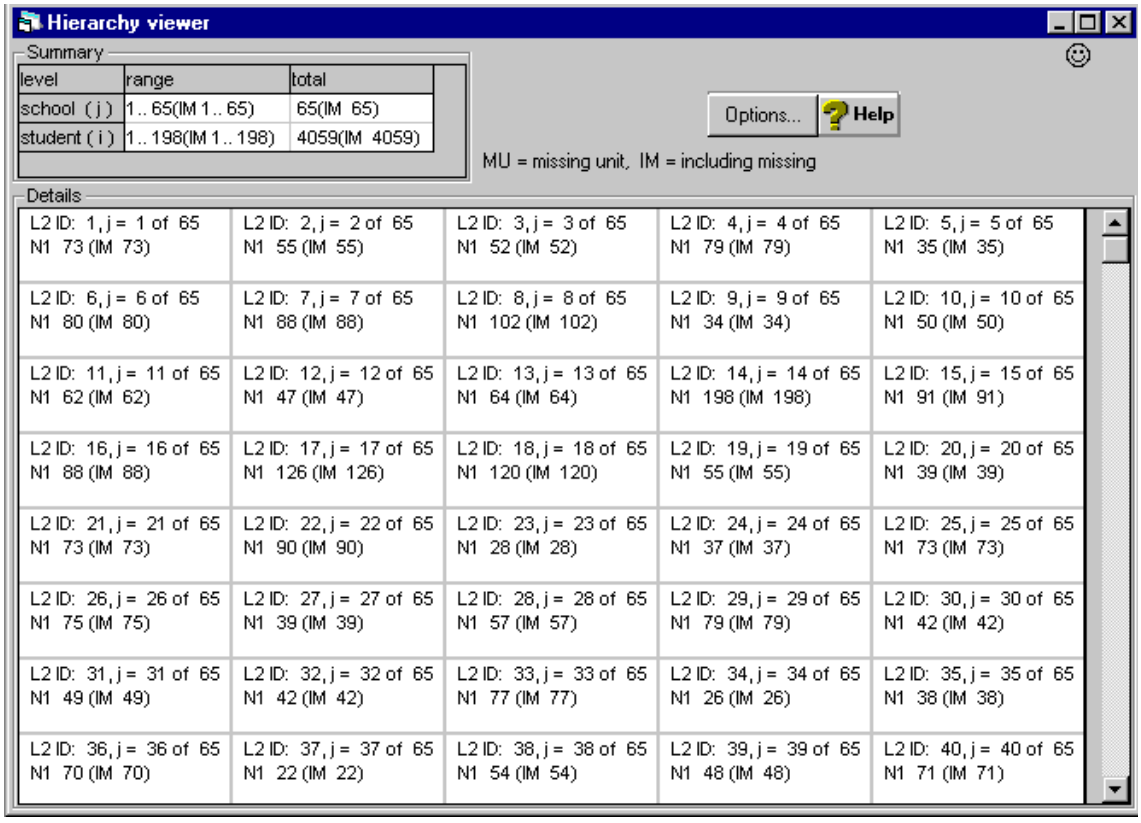
This display is somewhat verbose but a little more readable than the default subscript display. You can switch between the subscript formats by pressing the **subscripts** button. The screen shots in this chapter use the default subscript format. You can gain more control over how subscripts are displayed by clicking on **subscripts** from the model menu.

Before running a model it is always a good idea to get *MLwiN* to display a summary of the hierarchical structure to make sure that the structure *MLwiN* is using is correct. To do this

Select the **Model** menu

Select **Hierarchy Viewer**

Which produces:



The top **summary** grid shows, in the **total** column, that there are 4059 pupils in 65 schools. The range column shows that there are maximum of 198 pupils in any school. The **details** grid shows information on each school. 'L2 ID' means 'level 2 identifier value', so that the first cell under **details** relates to school no 1. If when you come to analyse your own data the hierarchy that is reported does not conform to what you expect, then the most likely reason is that your data are not sorted in the manner required by *MLwiN*. In an *n* level model *MLwiN* requires your data to be sorted by level 1, within level 2, within level 3...level *n*. There is a sort function available from the **Data Manipulation** menu.

We have now completed the specification phase for this simple model. It is a good idea to save the worksheet which contains the specification of the model so far, giving it a different name so that you can return to this point in the manual at a later time.

Estimation

We shall now get *MLwiN* to estimate the parameters of the model specified in the previous section.

We are going to estimate the two parameters β_0 and β_1 which in a single level model are the regression coefficients. In multilevel modelling regression coefficients are referred to as constituting the **fixed** part of the model**.** We also estimate the variance of the school level random effects σ^2_{u0} and the variance of the pupil level random effects σ^2_{e0}. The random effects and their variances are referred to as the **random** part of the model**.**

Click the **Estimates** button on the **Equations** window tool bar

You should see highlighted in blue the parameters that are to be estimated. Initially, we will not estimate the 4059 individual pupil level random effects and 65 school level random effects, we will return to these later.

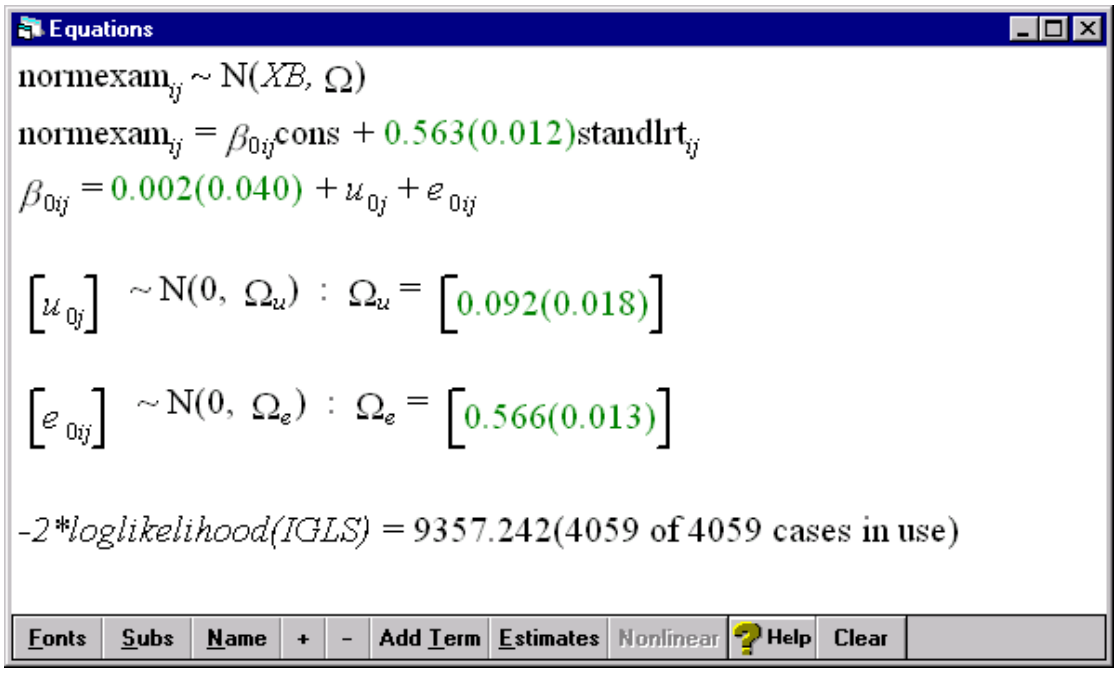
The estimation process is iterative. To begin the estimation we use the tool bar of the main *MLwiN* window. The **Start** button starts estimation, the **Stop** button stops it, and the **more** button resumes estimation after a stop. The default method of estimation is iterative generalised least squares (IGLS). This is noted on the right of the **Stop** button, and it is the method we shall use. The **Estimation control** button is used to vary the method, to specify convergence criteria, and so on. See the Help system for further details.

Click **Start**

You will now see the progress gauges at the bottom of the screen (R for random parameters and F for fixed parameters) fill up with green as the estimation proceeds alternately for the random and fixed parts of the model. In the present case this is completed at iteration 3 at which point the blue highlighted parameters in the **Equations** window change to green to indicate convergence. Convergence is judged to have occurred when all the parameters between two iterations have changed by less than a given tolerance, which is 10^{-2} by default but can be changed from the **Options** menu.

Click **Estimates**

once more and you will see the parameter estimates displayed together with their standard errors as in the following screen (the last line of the screen can be ignored for the time being).



The first two lines of this display reproduce equations 1.8, with the actual names of the different variables filled in. Recall that our model amounts to fitting a set of parallel straight lines to the results from the different schools. The slopes of the lines are all the same, and the fitted value of the common slope is 0.563 with a standard error of 0.012 (clearly, this is highly significant). However, the intercepts of the lines vary. Their mean is 0.002 and this has a standard error (in brackets) of 0.040. Not surprisingly with Normalized data, this is close to zero. The intercepts for the different schools are the level 2 residuals u_{0j} and these are distributed around their mean with a variance shown on line 4 of the display as 0.092 (standard error 0.018). The variance appears to be significantly different from zero. Judging significance for variances however, (and assigning confidence intervals) is not as straightforward as for the fixed part parameters. The simple comparison with the standard error and also the use of the **interval and tests** procedures (see help system) provides approximations that can act as rough guides. We shall deal with this further when discussing the likelihood ratio statistic and also in the part of this guide which deals with simulation based techniques. Of course, the actual data points do not lie exactly on the straight lines; they vary about them with amounts given by the level 1 residuals e_{0ij} and these have a variance estimated as 0.566, standard error 0.013. We shall see in the next chapter how *MLwiN* enables us to estimate and plot the residuals in order to obtain a better understanding of the model.

If we were to take children at random from the whole population, their variance would be the sum of the level 2 and level 1 variances, 0.092 + 0.566 = 0.658. The between-school variance makes up a proportion 0.140 of this total variance. This quantity is known as the *intra-school correlation*. It measures the extent to which the scores of children in the same school resemble each other as compared with those from children at different schools.

The last line of the display contains a quantity known as twice the *log likelihood*. This will prove to be useful in comparing alternative models for the data and carrying out significance tests. It can be ignored for the time being.

This is another place where you would do well to save the worksheet.

Graphing Predictions: Variance components

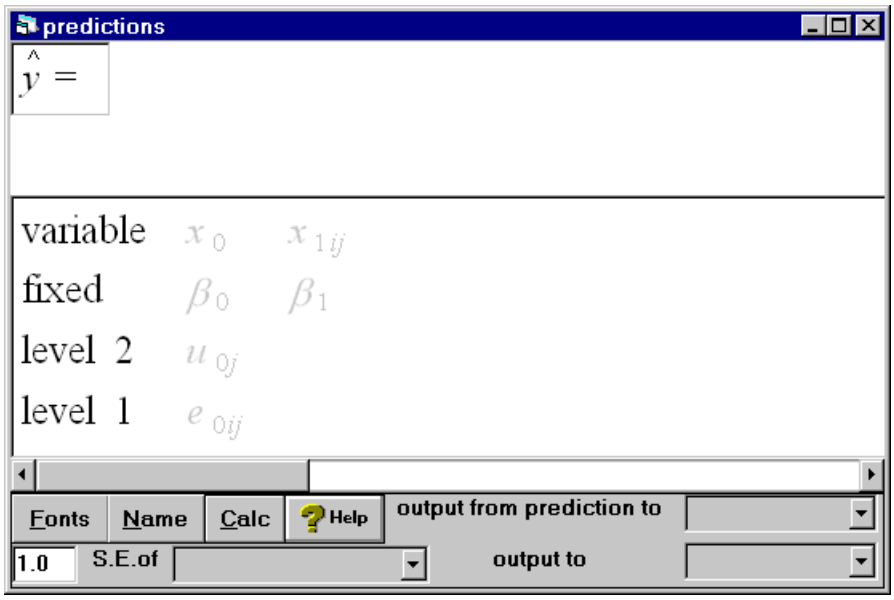
We have now constructed and fitted a *variance components* model in which schools vary only in their intercepts. It is a model of *simple* variation at level 2, which gives rise to the parallel lines illustrated in figure 1.2.

To demonstrate how the model parameters we have just estimated combine to produce the parallel lines of figure 1.2 we now introduce two new windows the **Predictions** window that can be used to calculate predictions from the model and the **Customised graphs** window which is a general purpose window for building graphs that can be used to graph our predicted values.

Lets start by calculating the average predicted line produced from the fixed part intercept and slope coefficients (β_0, β_1).

Select the **Model** menu Select **Predictions**

Which produces:

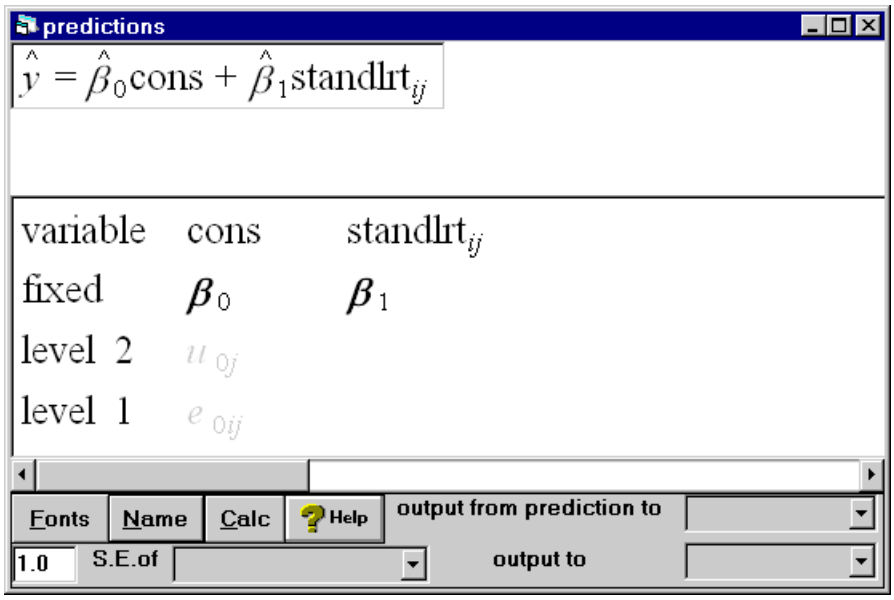


The elements of the model are arranged in two columns, one for each explanatory variable. Initially these columns are 'greyed out'. You build up a prediction equation in the top section of the window by selecting the elements you want from the lower section. Clicking on the variable name at the head of a column selects all the elements in that column. Clicking on an already-selected element deselects it.

Select suitable elements to produce the desired equation:

Click on β_0
Click on β_1
Click on **Names**

The **prediction** window should now look like this:



The only estimates used in this equation are $\hat{\beta}_0$ and $\hat{\beta}_1$, the fixed parameters – no random quantities have been included.

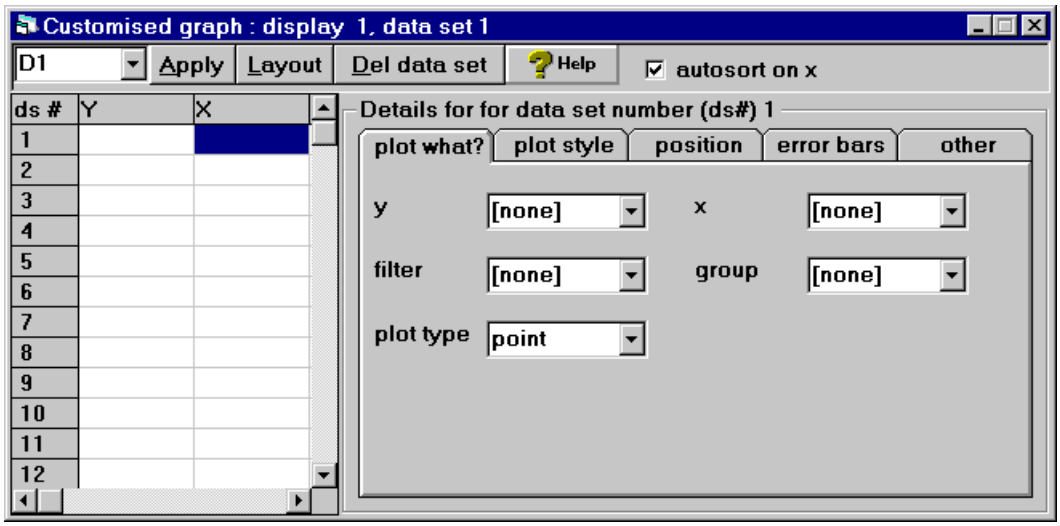
We need to specify where the output from the prediction is to go and then execute the prediction

In the **output from prediction to** drop-down list, select **C11** Click **Calc**

We now want to graph the predictions in column 11 against our predictor variable **standlrt**. We can do this using the **customised graph** window.

Select the **Graphs** menu Select **customised graph(s)**

This produces the following window:



This general purpose graphing window has a great deal of functionality, which is described in more detail both in the help system and in the next chapter of this guide. For the moment we will confine ourselves to its more basic functions. To plot out the data set of predicted values:

In the drop down list labelled **y** in the **plot what ?** tab select **c11**

In the neighbouring drop down list labelled **x** select **standlrt**

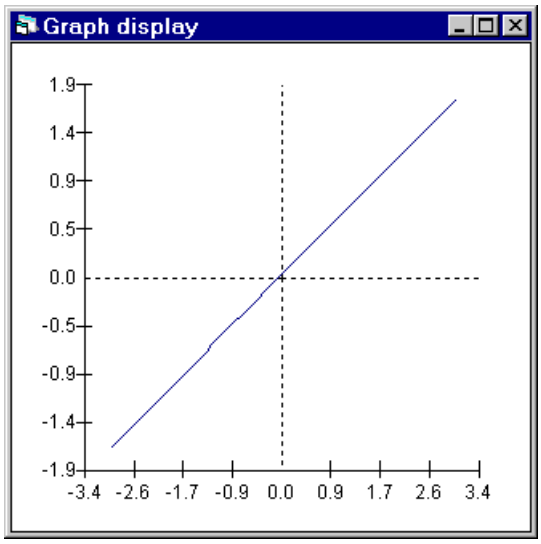
In the drop down list labelled **plot type** select **line**

In the drop down list labelled **group** select **school**

This last action specifies that the plot will produce one line for each school. For the present graph all the lines will coincide, but we shall need this facility when we update our predictions to produce the school level summary lines. To see the graph:

Click the **Apply** button

The following graph will appear:



We are now going to focus on the **predictions** window and the **graph** display window.

Close the **Equations** Close the **Customised graph** window

Arrange the **predictions** and graph display windows so that they are both visible. If by mistake you click on the interior area of the **graph display** window, a window offering advanced options will appear; if this happens just close the advanced options window; we will be dealing with this feature in the next chapter.

The line for the *j*'th school departs from the above average prediction line by an amount u_{0j}. The school level residual u_{0j} modifies the intercept term, but the slope coefficient β_1 is fixed. Thus all the predicted lines for all 65 schools must be parallel. To include the estimated school level intercept residuals in the prediction function:

Select the **predictions** window click on the term u_{0j}

The prediction equation in the top part of the **predictions** window changes from

$$\hat{y} = \hat{\beta}_0 \text{cons} + \hat{\beta}_1 \text{standlrt}_{ij}$$

to

$$\hat{y} = \hat{\beta}_{0j}\text{cons} + \hat{\beta}_1\text{standlrt}_{ij}$$

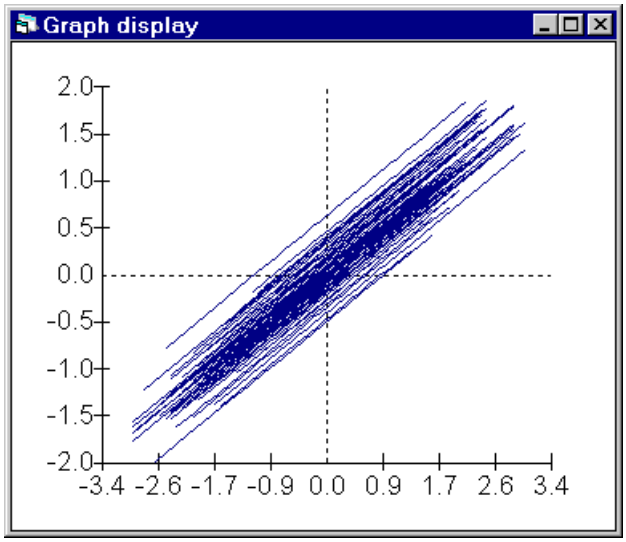
The crucial difference is that the estimate of the intercept $\hat{\beta}_0$ now has a j subscript. This subscript indicates that instead of having a single intercept, we have an intercept for each school, which is formed by taking the fixed estimate and adding the estimated residual for school j

$$\hat{\beta}_{0j} = \hat{\beta}_0 + \hat{u}_{0j}$$

We therefore have a regression equation for each school which when applied to the data produce 65 parallel lines. To overwrite the previous prediction in column 11 with the parallel lines

Press the **Calc** button in the prediction window

The graph display window is automatically updated with the new values in column 11 to show the 65 parallel lines.

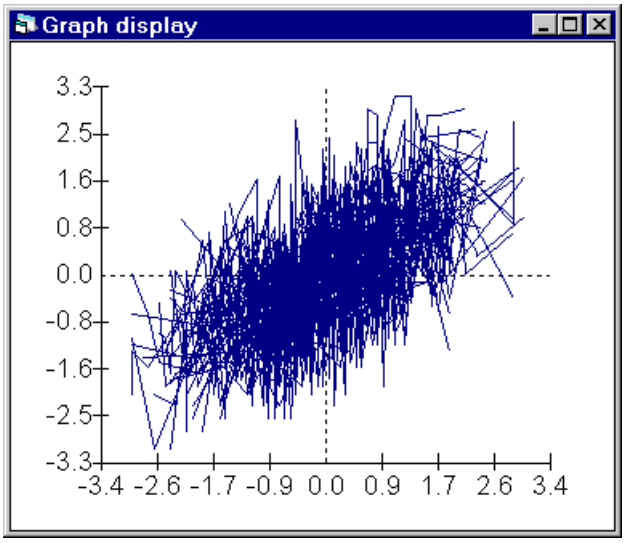


In this plot we have used the school level residuals (u_{0j}). Residuals and their estimation are dealt with in more detail in the next chapter.

Student i in school j departs from the school j summary line by an amount e_{0ij}. Recalculate the predictions to include e_{0ij} as well as u_{0j} as follows

Click on e_{0ij}

Press the **Calc** button



Which is a line plot through the original values of y_{ij}, i.e. we have predicted back onto the original data. Experiment including different combinations of $(\beta_0, \beta_1, u_{0j}, e_{0ij})$ in the prediction equation. Before pressing the **calc** button try and work out what pattern you expect to see in the graph window.

A Random slopes model

The *variance components* model which we have just specified and estimated assumes that the only variation between schools is in their intercepts. We should allow for the possibility that the school lines have different slopes as in Figure 1.3. This implies that the coefficient of standlrt will vary from school to school.

Still regarding the sample schools as a random sample from a population of schools, we wish to specify a coefficient of standlrt which is random at level 2. To do this we need to inform *MLwiN* that the coefficient of x_{1ij}, or $\mathbf{standlrt}_{ij}$, should have the subscript j attached.

To do this

Select the model menu Select the Equations window

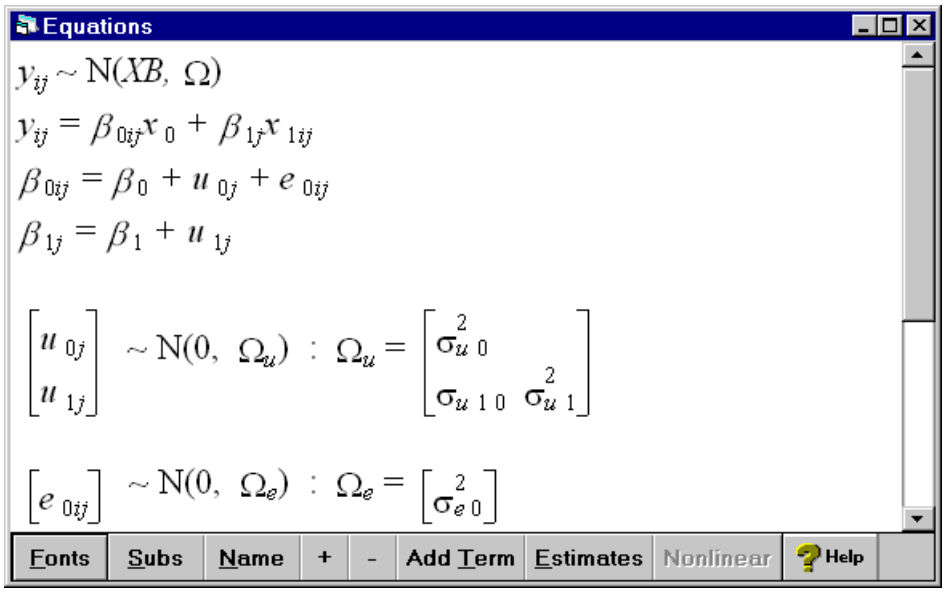
Click Estimates until β_0 etc. are displayed in black

Click β_1

Check the box labelled j (school)

Click **Done**

This produces the following result:



Now that the model is becoming more complex we can begin to explain the general notation. We have two explanatory variables x_0 and x_{1ij} (cons and standlrt). Anything containing a 0 subscript is associated with x_0 and anything containing a 1 subscript is associated with x_{1ij}. The letter *u* is used for random departures at level 2(in this case school). The letter *e* is used for random departures at level 1(in this case student).

The parameters β_0 and β_1 are the fixed part (regression coefficients) associated with x_0 and x_{1ij}. They combine to give the average line across all students in all schools.

The terms u_{0j} and u_{1j} are random departures or 'residuals' at the school level from β_0 and β_1. They allow the *j*'th school's summary line to differ from the average line in both its slope and its intercept.

The terms u_{0j} and u_{1j} follow a multivariate (in this case bivariate) Normal distribution with mean 0 and covariance matrix Ω_u. In this model we have two random variables at level 2 so Ω_u is a 2 by 2 covariance matrix. The elements of Ω_u are:

var (u_{0j}) = σ^2_{u0} (the variation across the schools' summary lines in their intercepts)

var (u_{1j}) = σ^2_{u1} (the variation across the schools' summary lines in their slopes)

cov (u_{0j} , u_{1j}) = σ_{u01} (the school level intercept/slope covariance).

Students' scores depart from their school's summary line by an amount e_{0ij}. (We associate the level 1 variation with x_0 because this corresponds to modelling constant or homogeneous variation of the student level departures. This requirement can be relaxed as we shall see later).

To fit this new model we could click Start as before, but it will probably be quicker to use the estimates we have already obtained as initial values for the iterative calculations. Therefore

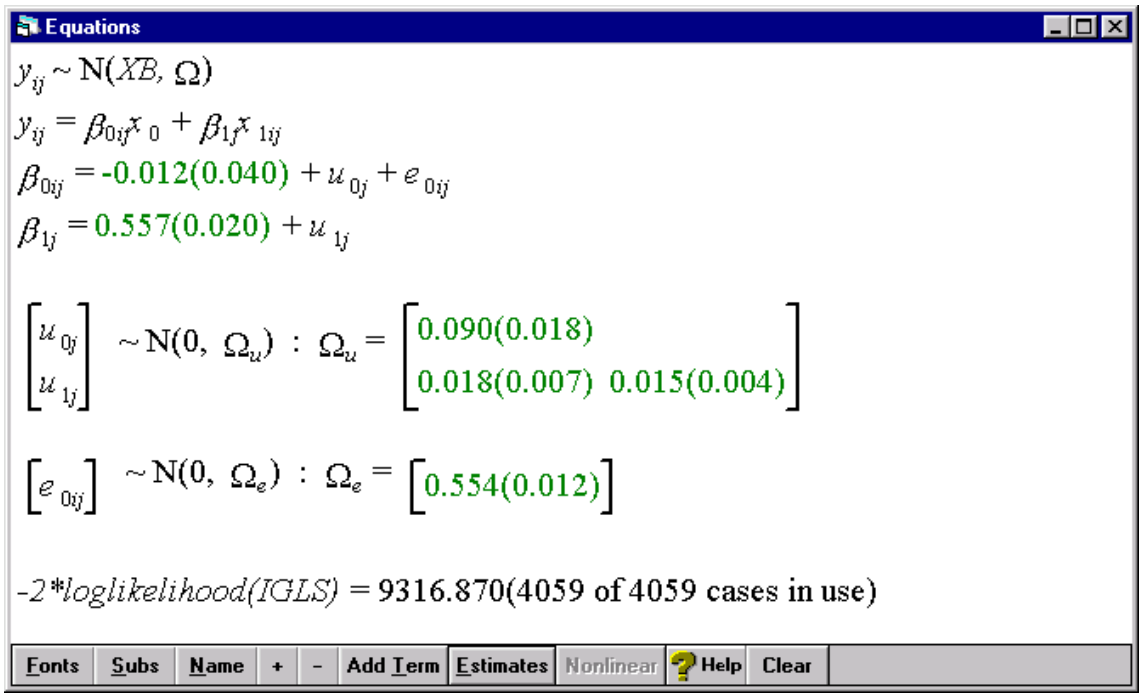
Click More

Convergence is achieved at iteration 7.

In order to see the estimates,

Click Estimates (twice if necessary) Click Names

To give



You should compare this display with that for the model where we did not fit a random slope. In line 2 of the display the coefficient of standlrt has acquired a suffix *j* indicating that it varies from school to school. In fact, its mean from line 4 is 0.557 (standard error 0.020), not far different from the model with a single slope. However, the individual school slopes vary about this mean with a variance estimated as 0.015 (standard error 0.004). The intercepts of the individual school lines also differ. Their mean is –0.012 (standard error 0.040) and their variance is 0.090 (standard error 0.018). In addition there is a positive covariance between intercepts and slopes estimated as +0.018 (standard error 0.007), suggesting that schools with higher intercepts tend to some extent to have steeper slopes and this corresponds to a correlation between the intercept and slope (across schools) of $0.018 / \sqrt{0.015 * 0.090} = 0.49$. This will lead to a fanning out pattern when we plot the schools predicted lines.

As in the previous model the pupils' individual scores vary around their schools' lines by quantities e_{0ij}, the level 1 residuals, whose variance is estimated as 0.554 (standard error 0.012).

The quantity on the last line of the display, -2*log-likelihood can be used to make an overall comparison of this more complicated model with the previous one. You will see that it has decreased from 9357.2 to 9316.9, a difference of 40.3. The new model involves two extra parameters, the variance of the slope residuals u_{1j} and their covariance with the intercept residuals u_{0j} and the change (which is also the change in deviance, where the deviance for Normal models differs by a constant term for a fixed sample size) can be regarded as a χ^2 value with 2 degrees of freedom under the null hypothesis that the extra parameters have population values of zero. As such it is very highly significant, confirming the better fit of the more elaborate model to the data.

Graphing predictions : random slopes

We can look at pattern of the schools summary lines by updating the predictions in the graph display window. We need to form the prediction equation

$$\hat{y} = \hat{\beta}_{0j} x_0 + \hat{\beta}_{1j} x_{1ij}$$

One way to do this is

Select the **Model** menu

Select **Predictions**

In the predictions window click on the word variables

From the menu that appears choose **Include all explanatory variables**

Click on e_{oij} to remove it from the prediction equation

In the **output from prediction to** drop-down list, select **c11**

Click **Calc**

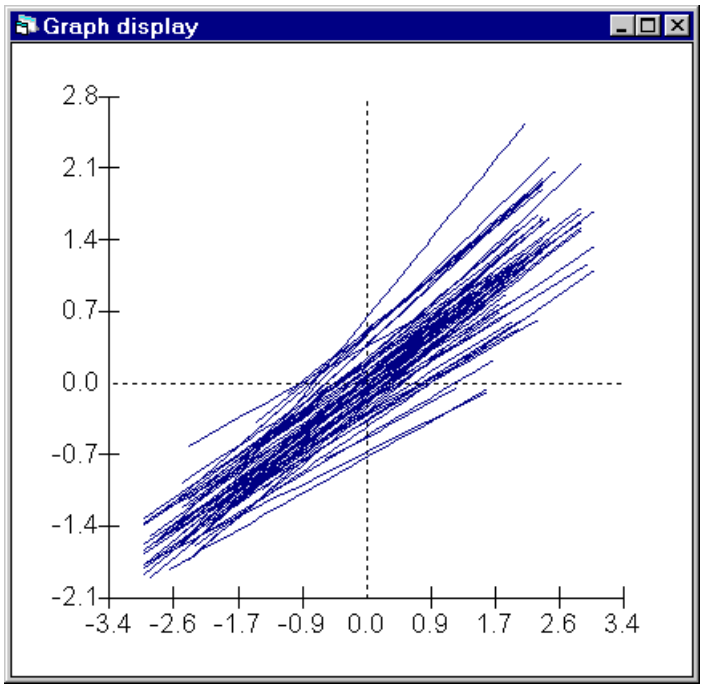
This will overwrite the previous predictions from the random intercepts model with the predictions from the random slopes model. The graph window will be automatically updated. If you do not have the graph window displayed, then

Select the **Graphs** menu

Select **customised graphs**

Click **Apply**

The graph display window should look like this :



The graph shows the fanning out pattern for the school prediction lines that is implied by the positive intercept/slope covariance at the school level.

To test your understanding try building different prediction equations in the **predictions** window; before you press the **calc** button try and work out how the graph in the **graph display** window will change.

That concludes the second chapter. It is a good idea to save your worksheet using the **save** option on the **File** menu.

What you should have learnt from this chapter

You should understand :

- What a random intercept model is
- What a random slope model is
- The equations used to describe these models
- How to construct, estimate and interpret these models using the equations window in *MLwiN*

- How to carry out simple tests of significance

- How to use the predictions window to calculate predictions from the model estimates

Chapter 3: Residuals

In this chapter we will work through the random slope model again. This time we shall explore the school and student random departures known as *residuals*.

Before we begin let's close any open windows :

Select the **Window** menu

Select **close all windows**

What are multilevel residuals?

In order to answer that question let's return to the random intercepts model. You can retrieve one of the earlier saved worksheets, or you can modify the random slopes model -

Select the **model** menu

Select **Equations**

Click on β_1

Uncheck the box labelled **j(school)**

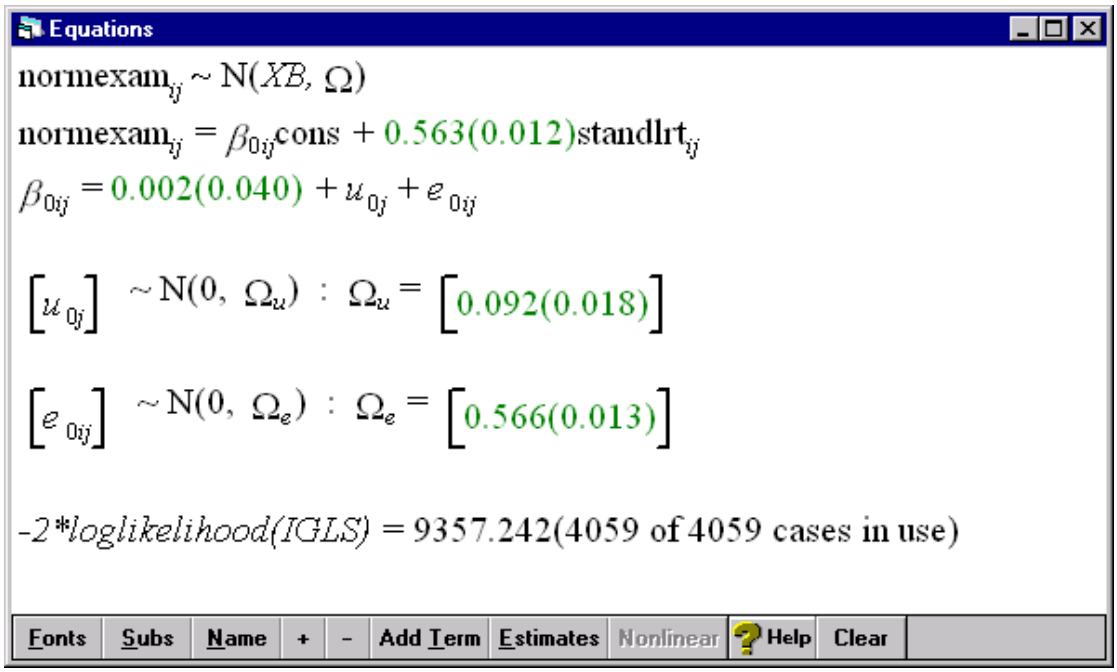
Click **Done**

The slope coefficient is now fixed with no random component. Now run the model and view the estimates:

Press **Start** on the main toolbar

Press **Name** then **Estimates** twice in the **Equations** window

Which produces :



This should be familiar from the previous chapter. The current model is a 2-level linear regression relationship of **normexam** on **standlrt**, with an average line defined by the two fixed coefficients β_0 and β_1. The model is made two-level by allowing the line for the *j*th school to be raised or lowered from the average line by an amount u_{0j}. These departures from the average line are known as the level 2 residuals. Their mean is zero and their estimated variance of 0.092 is shown in the **Equations** window. With educational data of the kind we are analysing, they might be called the school effects. In other datasets, the level 2 residuals might be hospital, household or area effects.

The true values of the level 2 residuals are unknown, but we will often require to obtain estimates of them. We might reasonably ask for the effect on student attainment of one particular school. We can in fact predict the values of the residuals given the observed data and the estimated parameters of the model (see Goldstein, 1995, Appendix 2.2). In ordinary multiple regression, we can estimate the residuals simply by subtracting the predictions for each individual from the observed values. In multilevel models with residuals at each of several levels, a more complex procedure is needed.

Suppose that y_{ij} is the observed value for the *i*th student in the *j*th school and that $\hat{y}_{ij}$ is the predicted value from the average regression line. Then the *raw residual* for this subject is $r_{ij} = y_{ij} - \hat{y}_{ij}$. The raw residual for the *j*th school is the mean of these over the students in the school. Write this as r_{+j}. Then the predicted level 2 residual for this school is obtained by multiplying r_{+j}. by a factor as follows –

$$\hat{u}_{0j} = \frac{\sigma^2{}_{u0}}{\sigma^2{}_{uo} + \sigma^2{}_{e0} / n_j} r_{+j}$$

where n_j is the number of students in this school.

The multiplier in the above formula is always less than or equal to 1 so that the estimated residual is usually less in magnitude than the raw residual. We say that the raw residual has been multiplied by a *shrinkage factor* and the estimated residual is sometimes called a shrunken residual. The shrinkage factor will be noticeably less than 1 when σ^2_{e0} is large compared to σ^2_{u0} or when n_j is small (or both). In either case we have relatively little information about the school (its students are very variable or few in number) and the raw residual is pulled in towards zero. In future 'residual' will mean shrunken residual. Note that we can now estimate the level 1 residuals simply by the formula

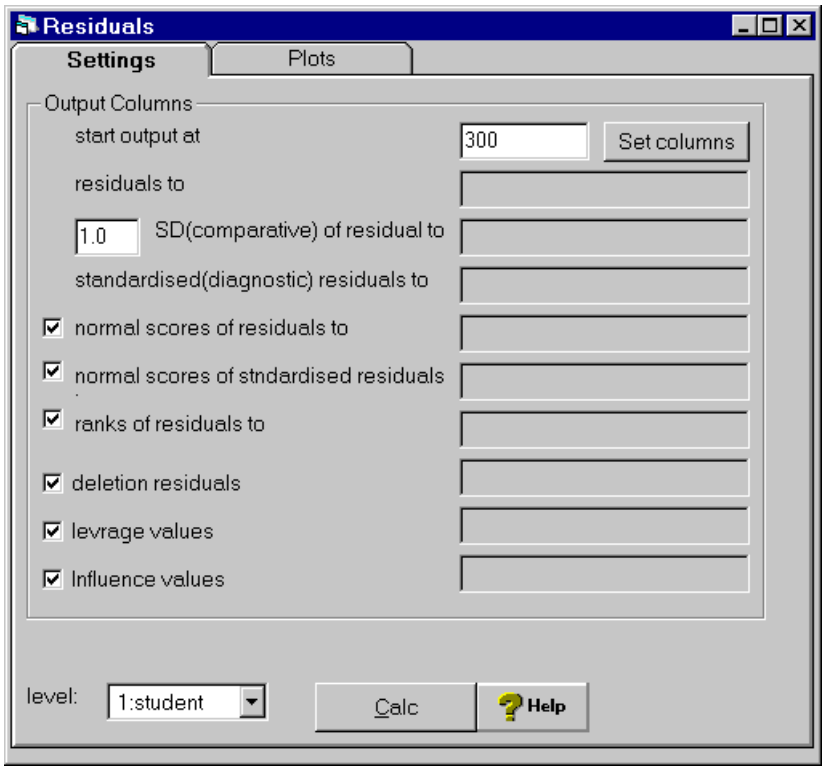
$$\hat{e}_{0ij} = r_{ij} - \hat{u}_{0j}$$

MLwiN is capable of calculating residuals at any level and of providing standard errors for them. These can be used for comparing higher level units (such as schools) and for model checking and diagnosis.

Calculating residuals in *MLwiN*

We can use the **Residuals** window in *MLwiN* to calculate residuals. Let's take a look at the level 2 residuals in our model.

Select **Model** menu
Select **Residuals**
Select **Settings** tab



The comparative standard deviation (SD) of the residual is defined as the standard deviation of $u_{0j} - \hat{u}_{0j}$ and is used for making inferences about the unknown underlying value u_{0j}, given the estimate $\hat{u}_{0j}$. The standardised residual is defined as $\hat{u}_{0j} / SD(\hat{u}_{0j})$ and is used for diagnostic plotting to ascertain Normality etc.

As you will see, this window permits the calculation of the residuals and of several functions of them. We need level 2 residuals, so at the bottom of the window

From the **level:** list **s**elect **2:school**

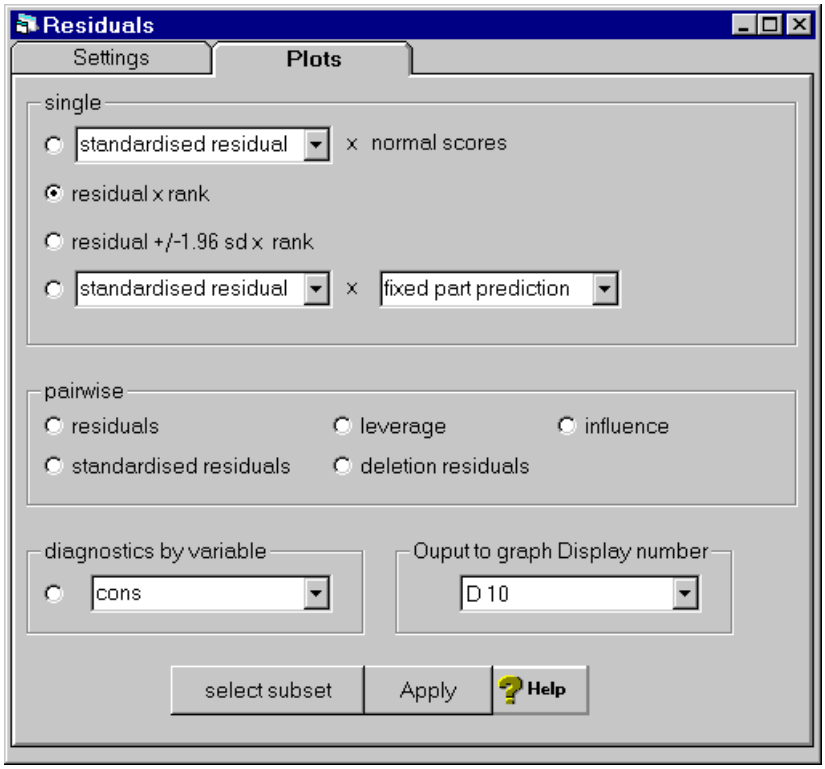
You also need to specify the columns into which the computed values of the functions will be placed.

Click the **Set columns** button

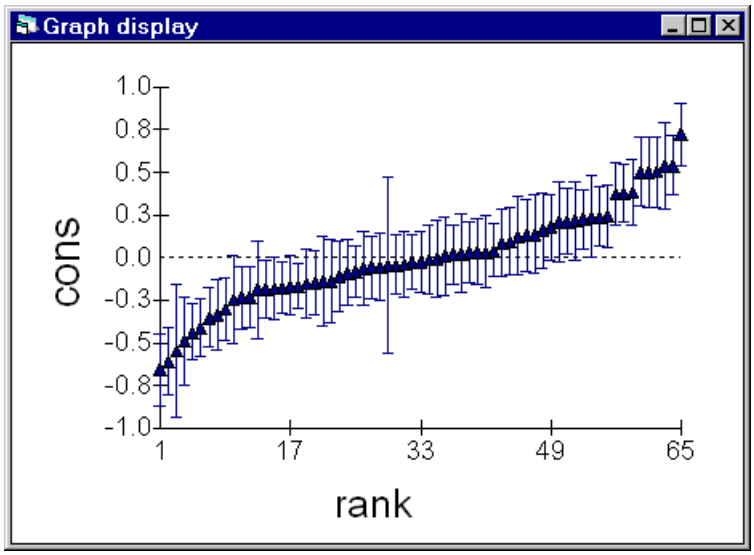
The nine boxes beneath this button are now filled in grey with column numbers running sequentially from C300. These columns are suitable for our purposes, but you can change the starting column by editing the **start output at** box. You can also change the multiplier to be applied to the standard deviations, which by default will be stored in C301.

Edit the SD multiplier to 1.96 Click **Calc(**to calculate columns C300 to C308.

Having calculated the school residuals, we need to inspect them and *MLwiN* provides a variety of graphical displays for this purpose. The most useful of these are available from the **Residuals** window by clicking on the **Plots** tab. This brings up the following window –



One useful display plots the residuals in ascending order with their 95% confidence limit. To obtain this, click on the third option in the **single** frame (residual +/- 1.96 SD x rank) then click **Apply**. The following graph appears



This is sometimes known (for obvious reasons) as a caterpillar plot. We have 65 level 2 residuals plotted, one for each school in the data set. Looking at the confidence intervals around them, we can see a group of 10 or 15 schools at each end of the plot where the confidence intervals for their residuals do not overlap zero. Remembering that these residuals represent school departures from the overall average line predicted by the fixed parameters, this means that the majority of the schools do not differ significantly from the average line at the 5% level.

See Goldstein and Healy (1995) for further discussion on how to interpret and modify such plots when multiple comparisons among level 2 units are to be made. Comparisons such as these, especially of schools or hospitals, raise difficult issues: in many applications, such as here, there are large standard errors attached to the estimates. Goldstein and Spiegelhalter (1996) discuss this and related issues in detail.

Note: You may find that you sometimes need to resize graphs in *MLwiN* to obtain a clear labelling of axes.

What you should have learnt from this chapter

- Multilevel residuals are shrunken towards zero and shrinkage increases as n_j decreases

- How to calculate residuals in *MLwiN*

Chapter 4. Graphical procedures for exploring the model

Displaying graphs

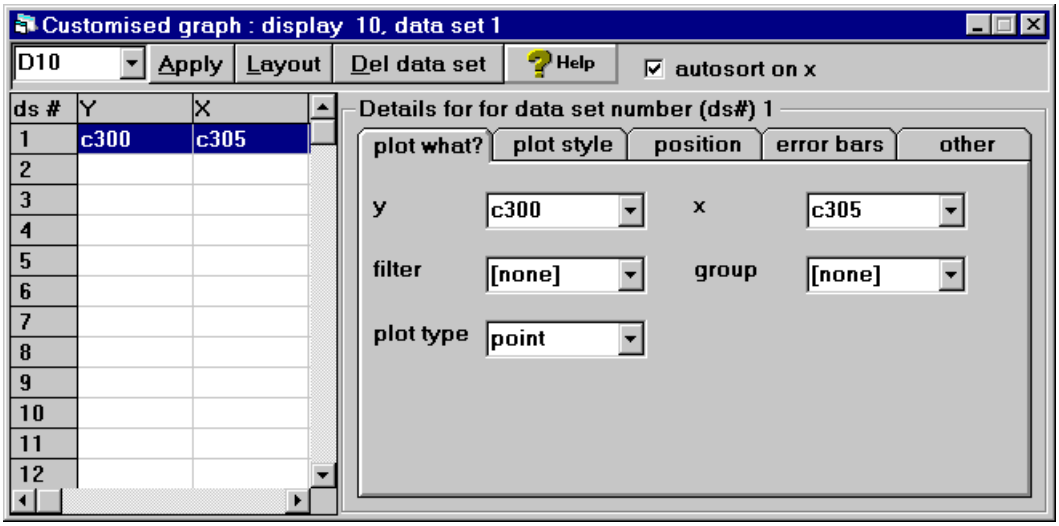
We have already produced a graphical display of the school level residuals in our random intercept model, using the **Residuals** window to specify what we wanted. *MLwiN* has very powerful graphical facilities, and in this chapter we shall see how to obtain more sophisticated graphs using the **Customised graphs** window. We will also use some of these graphical features to explore the random intercepts and random slopes models.

Graphical output in *MLwiN* can be described (very appropriately) at three levels. At the highest level, a *display* is essentially what can be displayed on the computer screen at one time. You can specify up to 10 different displays and switch between them as you require. A display can consist of several *graphs*. A graph is a frame with *x* and *y* axes showing lines, points or bars, and each display can show an array of up to 5x5 graphs. A single graph can plot one or more *datasets*, each one consisting of a set of *x* and *y* coordinates held in worksheet columns.

To see how this works,

Select the **graphs** menu Select **customised graphs**

The following window appears :



This screen is currently showing the construction details for display D10 – you may have noticed that the **plot** tab of the **Residuals** window in the previous chapter specified this in its bottom right hand corner. The display so far contains a single graph, and this in turn contains a single dataset, ds1 for which the *y* and *x* coordinates are in columns c300 and c305 respectively. As you can check from the **Residuals** window, these contain the level 2 residuals and their ranks.

Let us add a second graph to this display containing a scatterplot of *normexam* against *standlrt* for the whole of the data. First we need to specify this as a second dataset.

Select data set number 2 (**ds #2**) by clicking on the row labelled **2** in the grid on the left hand side of the window

Now use the *y* and *x* dropdown lists on the **plot what?** tab to specify **normexam** and **standlrt** as the *y* and *x* variables in ds2.

Next we need to specify that this graph is to separate from that containing the caterpillar plot. To do this,

Click the **position** tab on the right hand side of the **customised graph** window

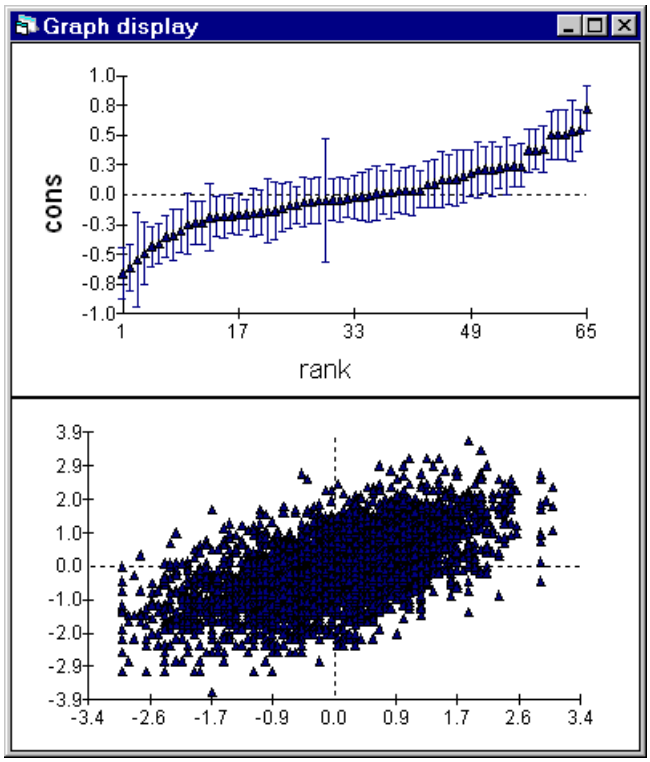
The display can contain a 5x5 grid or trellis of different graphs. The cross in the position grid indicates where the current data set, in this case (**normexam**, **standlrt**), will be plotted. The default position is row 1, column 1. We want the scatterplot to appear vertically below the caterpillar plot in row 2, column 1 of the trellis, so

Click the row 2 column 1 cell in the above grid

Now to see what we have got,

Press the **Apply** button at the top of the **Customised graph** window

and the following display will appear on the screen:



As a further illustration of the graphical facilities of *MLwiN*, let us create a third graph to show the 65 individual regression lines of the different schools and the average line from which they depart in a random manner. We can insert this between the two graphs that we already have. First we need to calculate the points for plotting in the new graph. For the individual lines

Select the **Model** window

Select **Predictions**

Click on **Variable**

Select **Include all explanatory variables**

Click on e_{0ij} to remove it

In the **output from prediction** list select c11

Press **calc**

This will form the predictions using the level 2 (school)residuals but not the level 1 (student) residuals. For the overall average line we need to eliminate the level 2 residuals, leaving only the fixed part of the model:

In the **Predictions** window click on u_{0j} to remove it

In the **output from prediction** list select c12

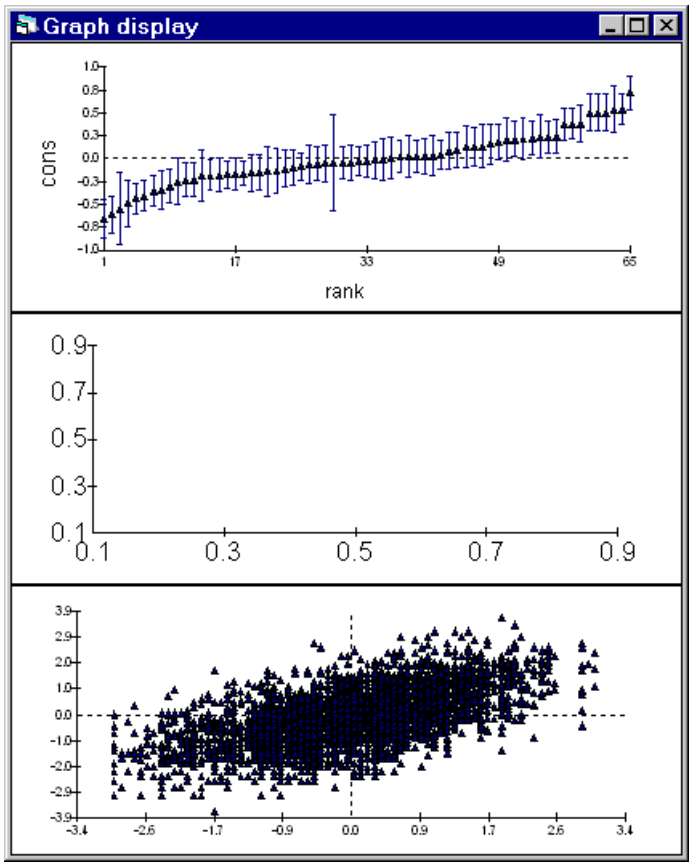
Press **calc**

Close the **Predictions** window

The **Customised graph** window is currently showing the details of dataset ds2, the scatterplot. With this dataset selected

Click on the **position** tab In the grid click the cell in row 3, column 1 Press **Apply**

The display now appears as follows:

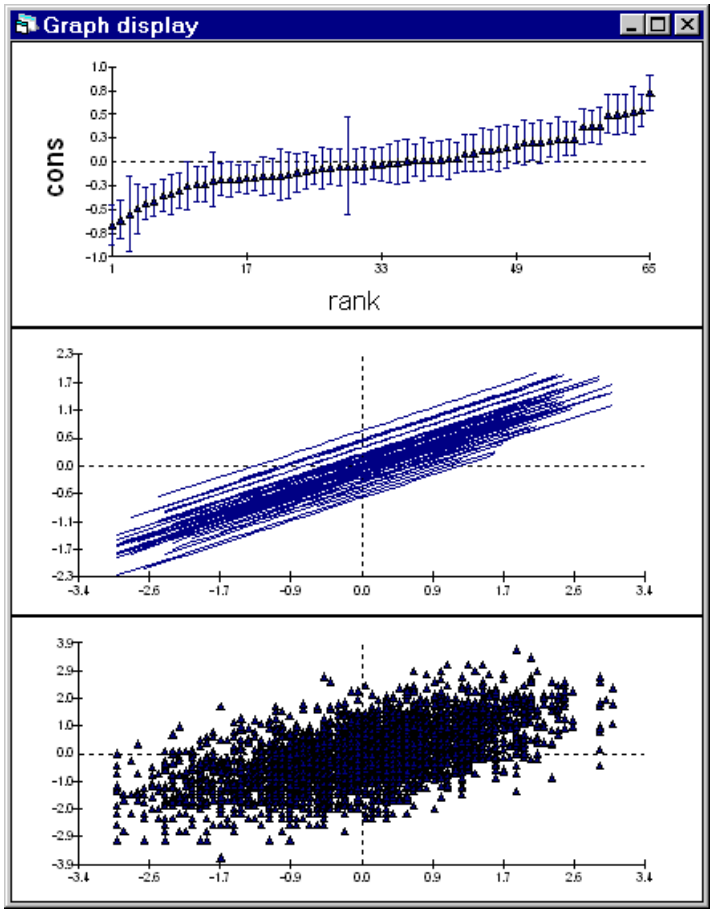


We have not yet specified any datasets for the middle graph so it is blank for the time being. Here and elsewhere you may need to resize and re-position the graph display window by pulling on its borders in the usual way.

Now let us plot the lines that we have calculated. We need to plot c11 and c12 against *standlrt*. For the individual school lines we shall need to specify the group, meaning that the 65 lines should be plotted separately. In the Customised graphs window

Select data set ds3 at the left of the window
In the **y** dropdown list specify c11
In the **x** dropdown list specify *standlrt*
In the **group** dropdown list select *school*
In the **plot type** dropdown list select *line*
Select the **position** tab
In the grid click the cell in row 2 column 1
Click **Apply**

This produces the following display:



Now we can superimpose the overall average line by specifying a second dataset for the middle graph. So that it will show up, we can plot it in red and make it thicker than the other lines:

Select dataset ds4 at the left hand side of the **Customised graphs** window

In the **y** dropdown list select c12

In the **x** dropdown list select *standlrt*

In the **plot type** dropdown list select *line*

Select the **plot styles** tab

In the **colour** dropdown list select *red*

In the **line thickness** dropdown list select 2

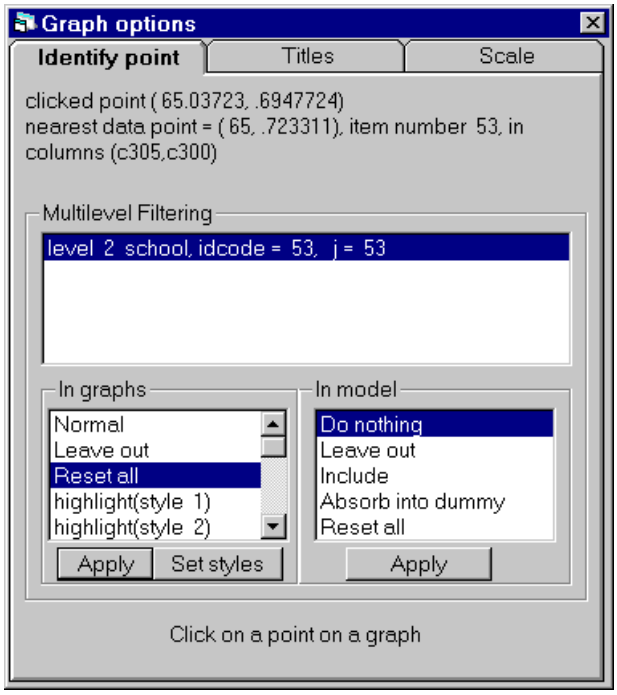
Select the **position** tab

In the grid click the cell in row 2, column 1

[I think the position should be OK following the previous manoeuvre]

Click **Apply**

There is a lot more that *MLwiN* makes it possible to do with the graphs that we have produced. To investigate some of this, click in the top graph on the point corresponding to the largest of the level 2 residuals, the one with rank 65. This brings up the following **Graph options** screen:



The box in the centre shows that we have selected the 53rd school out of the 65, whose identifier happens to be 53. We can *highlight* all the points in the display that belong to this school by selecting **highlight (style 1)** and clicking **Apply**. If you do this you will see that the appropriate point in the top graph, two lines in the middle graph and a set of points in the scatterplot have all become coloured red.

The individual school line is the thinner of the two highlighted lines in the middle graph. As would be expected from the fact that it has the highest intercept residual, the school's line is at the top of the collection of school lines.

It is not necessary to highlight all references to school 53. To de-highlight the school's contribution to the overall average line which is contained in dataset ds4, in the **Customised graphs** window:

Select dataset 3

Click on the **other** tab

Click the Exclude from highlight box

Click **Apply**

In the caterpillar plot there is a residual around rank 30 which has very wide error bars. Let us try to see why. If you click on the point representing this school in the caterpillar plot, the **graph options** window will identify it as school 48. Highlight the points belonging to this school in a different colour:

Using the graph options window, in the in graphs box select highlight (style 2)

Click **Apply**

The points in the scatterplot belonging to this school will be highlighted in cyan, and inspection of the plot shows that there are only two of them. This means that there is very little information regarding this school. As a result, the confidence limits for its residual are very wide, and the residual itself will have been shrunk towards zero by an appreciable amount.

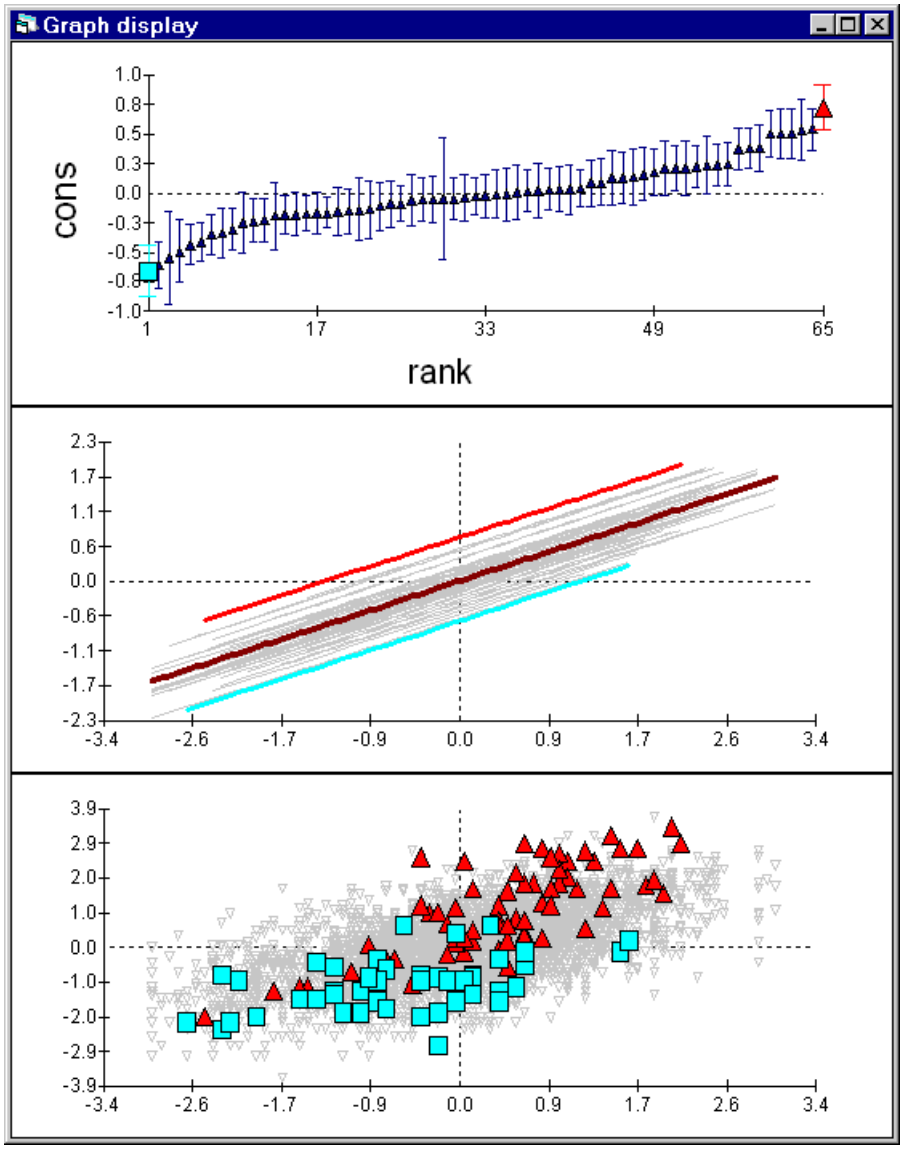
Next let us remove all the highlights from school 48. In the **graph options** window

In the **in graphs** box select **normal**

Click **Apply**

Now let us look at the school at the other end of the caterpillar, that with the lowest school level residual. Click on its point in the caterpillar (it turns out to be school 59) and in the **Graph options** window select **highlight (style 3)** and click **Apply**. The

highlighting will remain and the graphical display will look something like this, having regard to the limitations of monochrome reproduction:



The caterpillar tells us simply that school 59 and 53 have different intercepts – one is significantly below the average line, the other significantly above it. But the bottom graph suggests a more complicated situation. At higher levels of *standlrt*, the points for school 53 certainly appear to be consistently above those for school 59. But at the other end of the scale, at the left of the graph, there does not seem to be much difference between the schools. The graph indeed suggests that the two schools have different slopes, with school 53 the steeper.

To follow up this suggestion, let us keep the graphical display while we extend our model to contain random slopes. To do this:

From the **Model** menu select **Equation**

Click on β_1 and check the box labelled *j (school)* to make it random at level 2

Click **Done**

Click **More** on the main toolbar and watch for convergence

Close the **Equations** window

Now we need to update the predictions in column c11 to take account of the new model:

From the **Model** menu select **Predictions**

Click on u_{0j} and u_{1j} to include them in the predictions

In the **Output from predictions** dropdown list select c11

Click **Calc**

Notice that the graphical display is automatically updated with the new contents of column c11.

The caterpillar plot at the top of the display however is now out of date, having been calculated from the previous model. (Recall we used the residuals window to create the caterpillar plot). We now have two sets of level 2 residuals, one giving the intercepts for the different schools and one the slopes. To calculate and store these:

Select **Residuals** from the **Model** menu

Select **2:School** from the **level** dropdown list

Edit the **Start output at** box to 310

Click **Calc**

The intercept and slope residuals will be put into columns c310 and c311. To plot them

against each other:

In the **Customised graphs** window select dataset ds#1 and click **Delete dataset**

From the *y* dropdown list select c310

From the *x* dropdown list select c311

Click **Apply**

The axis titles in the top graph also need changing. Note that if you use the customised graph window to create graphs no titles are automatically put on the graphs. This is because a graph may contain many data sets so in general there is no obvious text for the titles. The existing titles appear because the graph was originally constructed by using the **plots** tab on the **residuals** window. You can specify or alter titles by clicking on a graph. In our case:

Click somewhere in the top graph to bring up the **Graph options** window

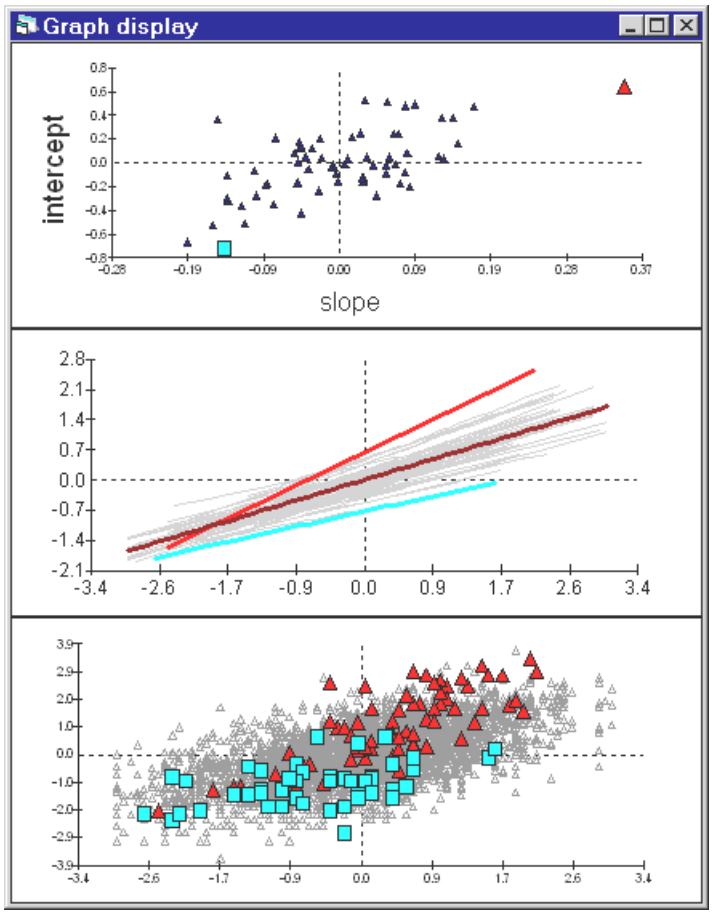
Select the **titles** tab

Edit the *y title* to be *Intercept*

Edit the *x title* to be *Slope*

Click **Apply**

You can add titles to the other graphs in the same way if you wish. Now the graphical display will look like this:


Graph display
intercept
slope
0.8
0.6
0.4
0.2
0.0
-0.2
-0.4
-0.6
-0.8
-0.28
-0.19
-0.09
0.00
0.09
0.19
0.28
0.37
2.8
2.1
1.4
0.7
0.0
-0.7
-1.4
-2.1
-3.4
-2.6
-1.7
-0.9
0.0
0.9
1.7
2.6
3.4

The two schools at the opposite ends of the scale are still highlighted, and the middle graph confirms that there is very little difference between them at the lower levels of **standlrt**. School 53 stands out as exceptional in the top graph, with a high intercept and much higher slope than the other schools.

For a more detailed comparison between schools 53 and 49, we can put 95% confidence bands around their regression lines. To calculate the widths of the bands:

Select **Predictions** from the **Model** menu

Edit the multiplier of **S.E.** to 1.96

From the **S.E. of** dropdown list select *level 2 resid function*

From the **output to** dropdown list select column c13

Click **Calc**

Now plot the bands

In the **Customised graphs** window select dataset ds#2

Select the **errors** tab

From the **y error +** list select c13

From the **y error –** list select c13

From the **y error type** list select *lines*

Click **Apply**

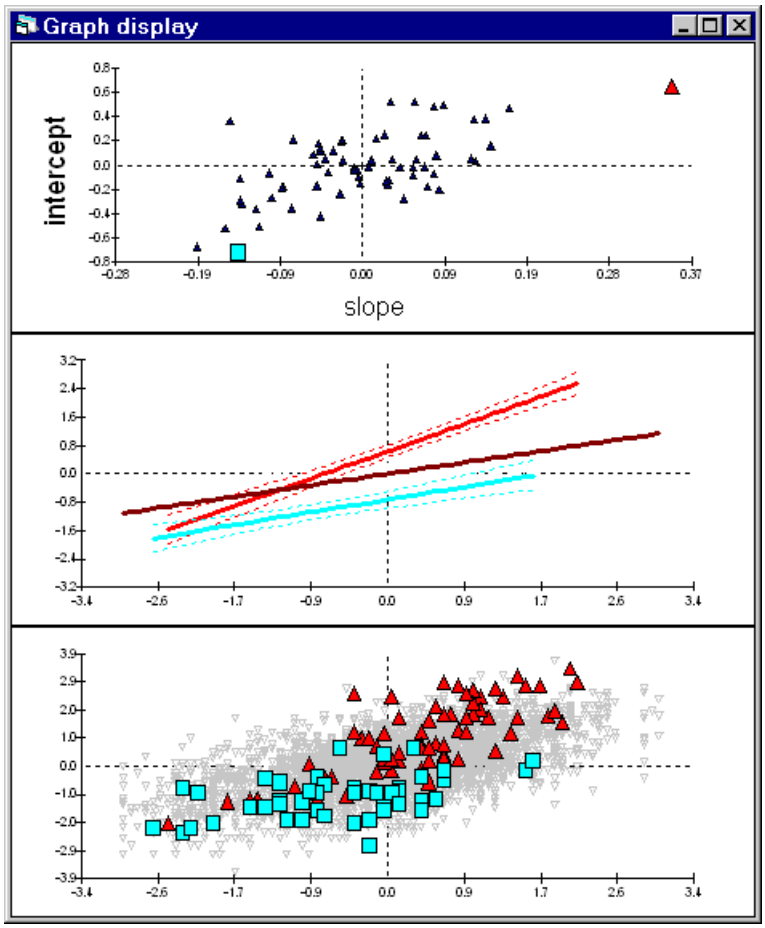
This draws 65 confidence bands around 65 school lines, which is not a particularly readable graph. However, we can focus in on the two highlighted schools by drawing the rest in white.

Select the **customised graphs** window

Select data set number 2 (**ds # 2**)

From the **colour** list, on the **plot style** tab, select **white**

Click Apply



The confidence bands confirm that what appeared to be the top and bottom schools cannot be reliably separated at the lower end of the intake scale.

Looking at the intercepts and slopes may be able to shed light on interesting educational questions. For example, schools with high intercepts and low slopes, plotting in the top left quadrant of the top graph, are 'levelling up' – they are doing well by their students at all levels of initial ability. Schools with high slopes are differentiating between levels of intake ability. The highlighting and other graphical features of *MLwiN* can be useful for exploring such features of complicated data. See Yang et al., (1999) for a further discussion of this educational issue.

What you should have learnt from this chapter

How to make different graphical representations of complex data.

How to explore aspects of multilevel data using graphical facilities such as highlighting.

With random slopes models differences between higher level units (e.g. schools) can not be expressed by a single number.

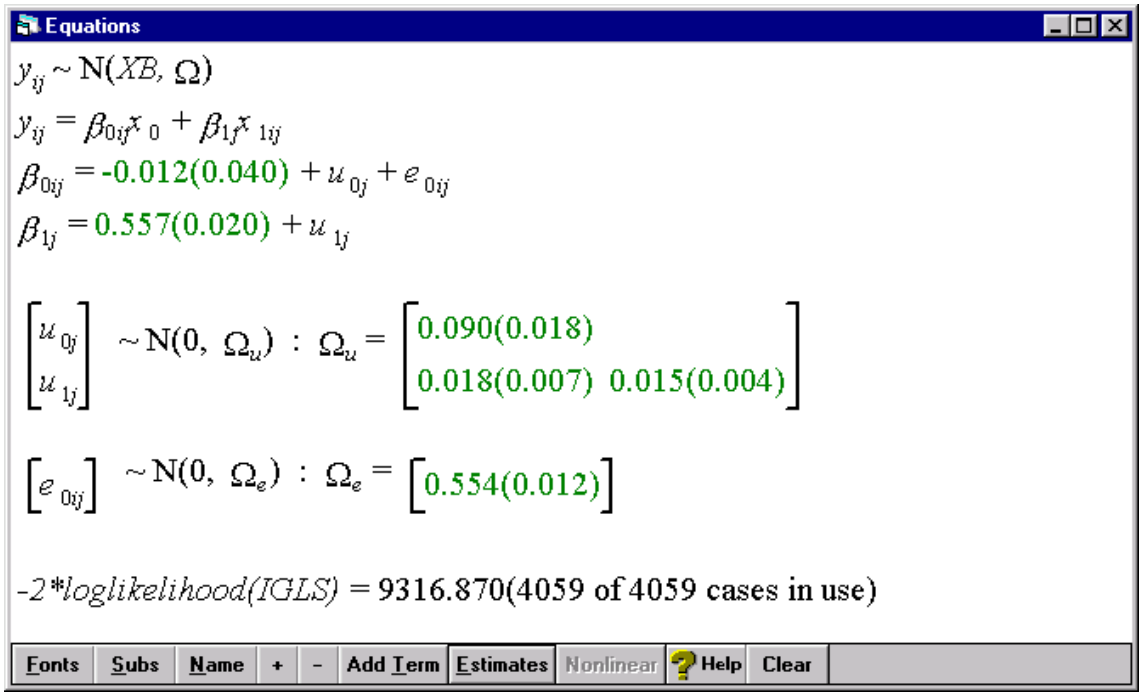
Chapter 5: Contextual effects

Many interesting questions in social science are of the form how are individuals effected by their social contexts? For example,

- Do girls learn more effectively in a girls' school or a mixed sex school?
- Do low ability pupils fare better when they are educated alongside higher ability pupils or worse?

In this section we will develop models to investigate these two questions.

First, here are the estimates we obtained at the end of chapter 2:



Before we go on let's close all open windows by

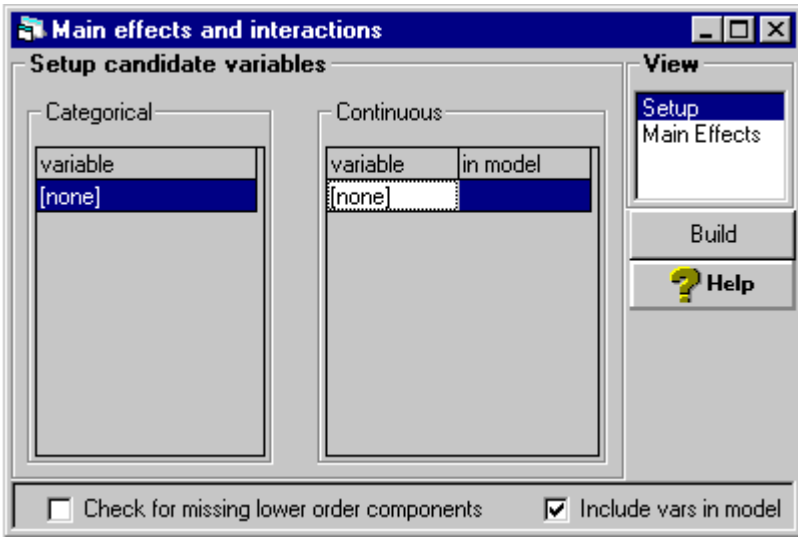
Select the **Window** menu Select **close all windows**

Pupil gender and school gender effects

We are now going use a new window which is useful for building models with categorical explanatory variables.

Select the **Model** menu Select **Main Effects and Interactions**

The following window appears



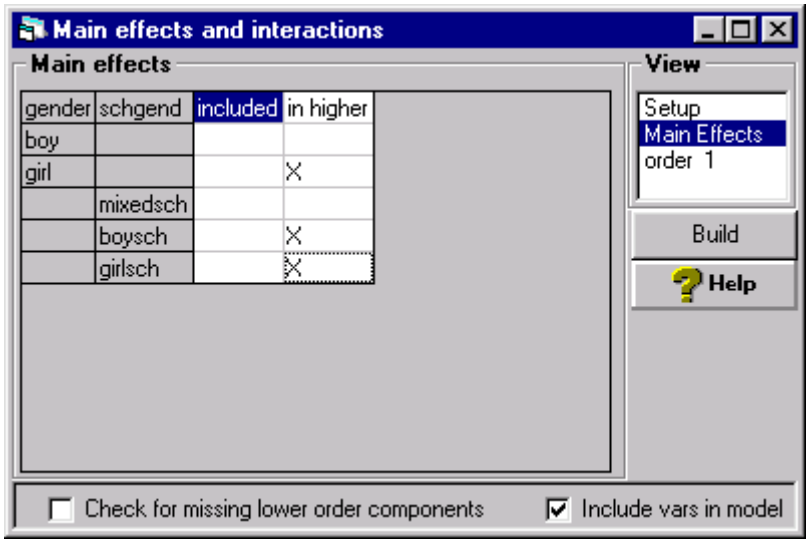
This screen automates the process of creating sets of dummy variables (and interactions between sets of dummy variables) that are required for modelling categorical predictors. To enter main effects for individual gender and school gender

In the panel marked **categorical** click on **[none]** From the list that appears select **gender** Click on **[none]** again and select **schgend**

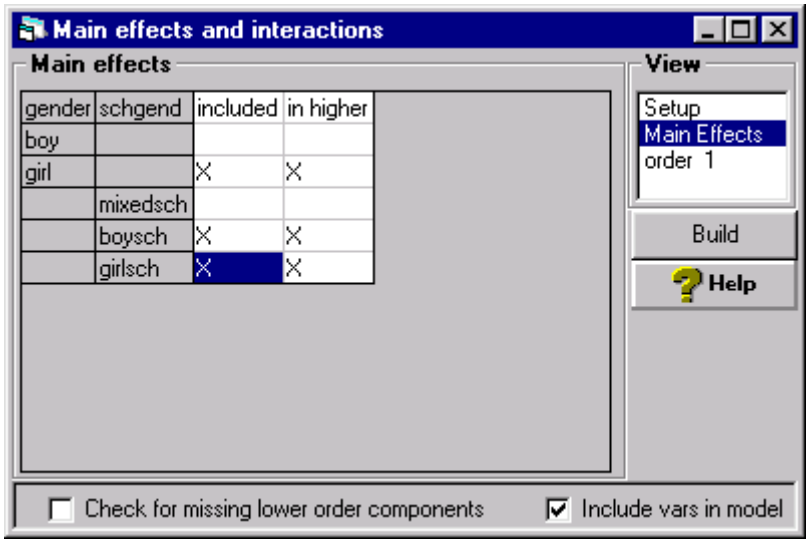
Note that now that we have defined two categorical variables the possibility for a 1st order interaction exists and the **view** panel (on the right of the window) has been updated to include the option **order 1.** We just want to include main effects so

In the **View** panel click on **Main Effects**

The main effects and interactions window now displays a list of potential main effects:



At the moment no main effects are included, to fit gender and school gender with **boy** and **mixedsch** as the reference categories, click on the corresponding entries in the **included** column to produce the pattern :



The **in higher** column defines what categories are made available for higher order interactions. This is useful when you have large numbers of categorical variables and the number of possible combinations for higher order interactions is very large.

To add the main effects to the model :

Click **Build**

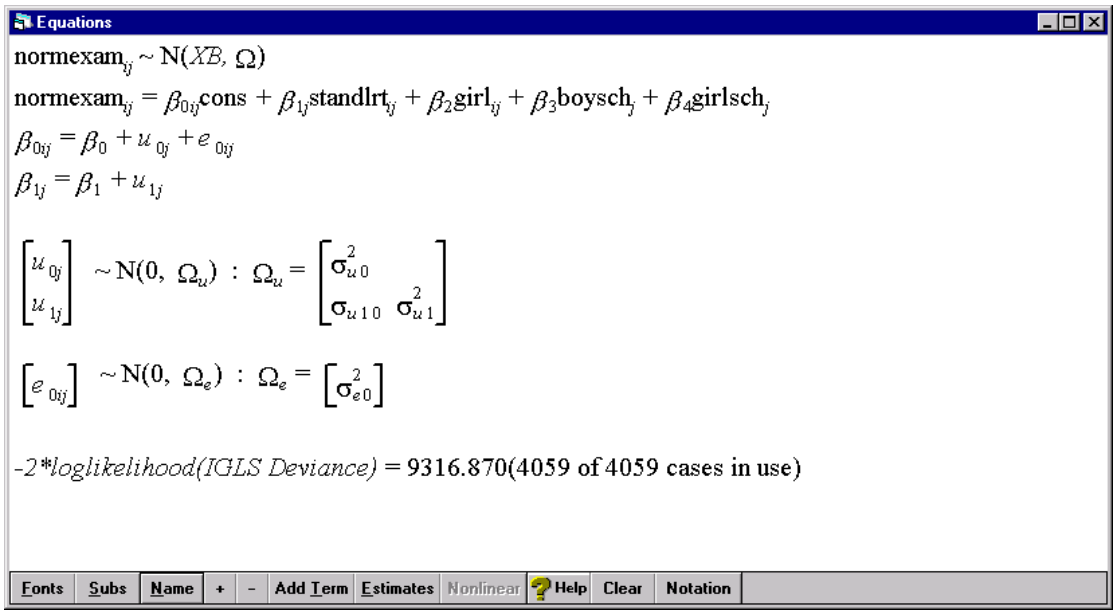
To view the model

Select the **Model** menu

Select **Equations**

Click **Names**

Which produces the following model

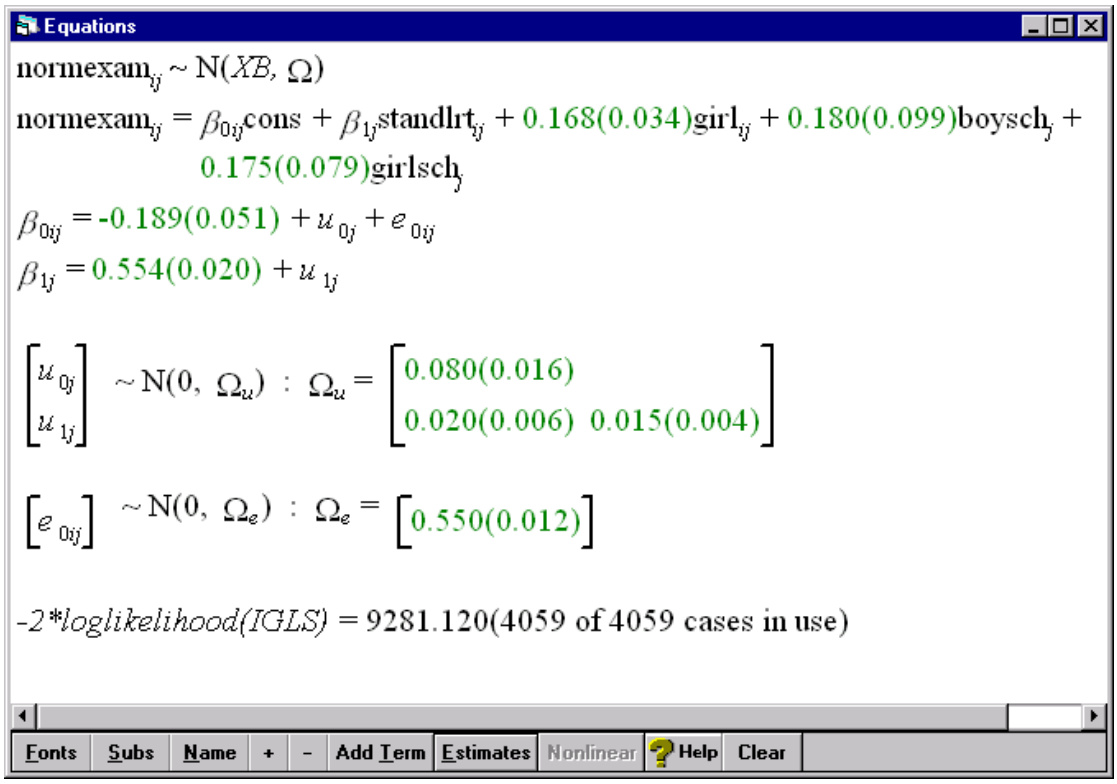


You can see that main effects for **girl, boysch** and **girlsch** have been added. **Girl** has subscript *ij* because it is a pupil level variable, whereas the two school level variables have subscript *j*. We can run the model and view the results by

Click **estimates** until numbers appear in the equations window

Press **More** on the main toolbar

The model converges to the results below:



The reference categories are boys in a mixed school. Girls in a mixed school do 0.168 of a standard deviation better than boys in a mixed school. Girls in a girls school do 0.175 points better than girls in a mixed school and (0.175+0.168) points better than boys in a mixed school. Boys in a boys school do 0.18 points better than boys in a mixed school.

Adding these three parameters produced a reduction in the deviance of 35, which, under the null hypothesis of no effects, follows a chi-squared distribution with three degrees of freedom. You can look this probability up using the **Tail Areas** option on the **Basic Statistics** menu. The value is highly significant.

In the 2 by 3 table of gender by school gender there are two empty cells, there are no boys in a girls school and no girls in a boys school. We are currently using a reference group and three parameters to model a four entry table, therefore because of the empty cells the model is saturated and no higher order interactions can be added.

The pupil gender and school gender effects modify the intercept (**standlrt**=0). An interesting question is do these effects change across the intake spectrum. To address this we need to extend the model to include the interaction of the continuous variable **standlrt** with our categorical variables. To do this

Select the **Main Effects and Interactions** window

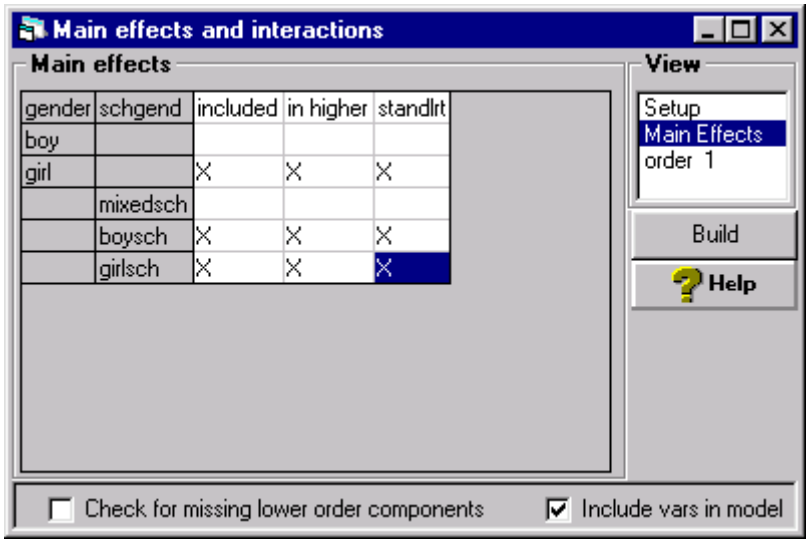
In the **View** panel select **setup**

In the **continuous** panel click on **none**

From the list that appears select **standlrt**

In the **View** panel select **main effects**

The main effects screen now has a column for **standlrt** added. Click on entries in the column to produce the following pattern



Click on **Build**

The equations window will be automatically modified to include the three new interaction terms. Run the model :

press **More** on the main toolbar

The deviance reduces by less than one unit. From this we conclude there is no evidence of an interaction between the gender variables and intake score. We can remove from the model by

Select the **main effects and interactions** window

Ensure **main effects** are selected in the **view** panel

Deselect all entries marked with a **X** in the standlrt column by clicking

Press **Build**

Note that we could have clicked on individual terms in **Equations** window and selected the **delete term** option. However, this would not have removed the terms from the main effects and interactions tables and every subsequent **build** would put them back into the model.

Contextual effects

The variable **schav** is constructed by taking the average intake ability(**standlrt**) for each school, based on these averages the bottom 25% of schools are coded 1(**low**), the middle 50 % coded 2(**mid**) and the top 25% coded 3(**high**). Let's include this categorical school level contextual variable in the model.

Select the **Main Effects and Interactions** window

Select S**etup** from the **View** panel

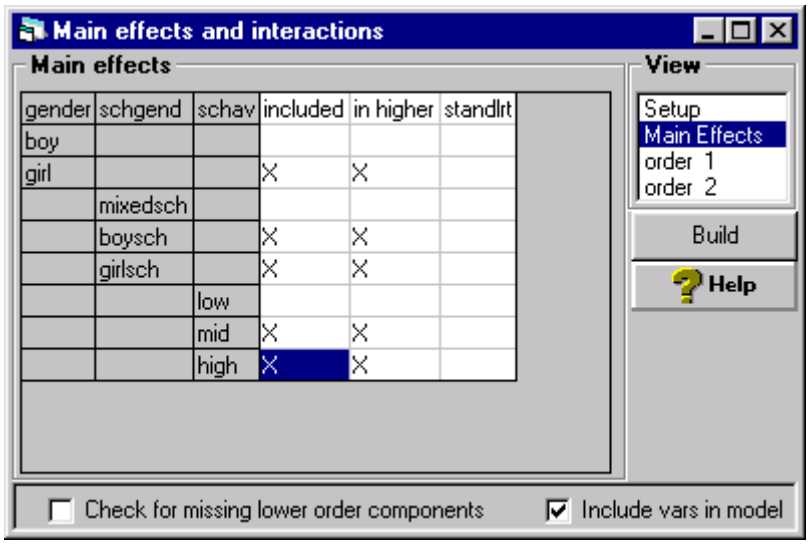
In the **Categorical** panel click on **[none]**

Select **schav** from the list that appears

Select **Main Effects** from the **View panel**

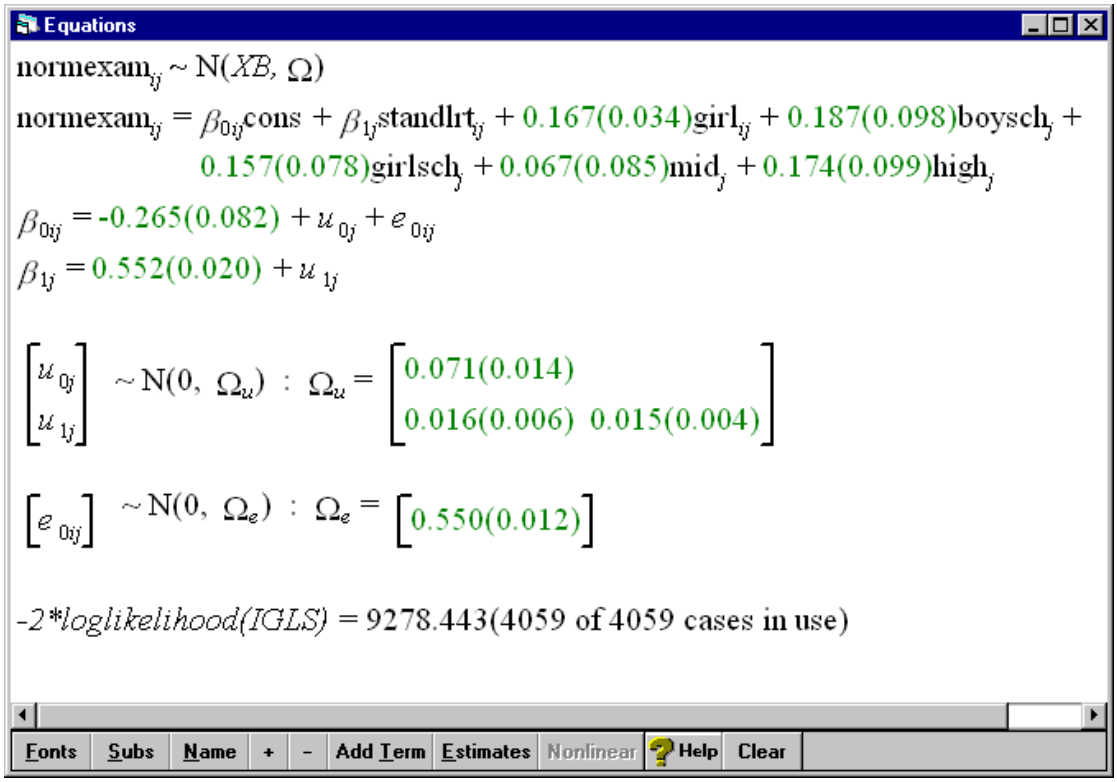
Click the **included** column for the **mid** and **high** entries

The main effects and interactions window should now look like this :



Click **Build**

Run the model by pressing **more** on the main toolbar



Children attending **mid** and **high** ability schools score 0.067 and 0.174 points more than children attending **low** ability schools. The effects are of borderline statistical significance. This model assumes the contextual effects of school ability are the same across the intake ability spectrum because these contextual effects are modifying the intercept term. That is the effect of being in a **high** ability school is the same for low ability and high ability pupils. To relax this assumption we need to include the interaction between **standlrt** and the school ability contextual variables. To do this :

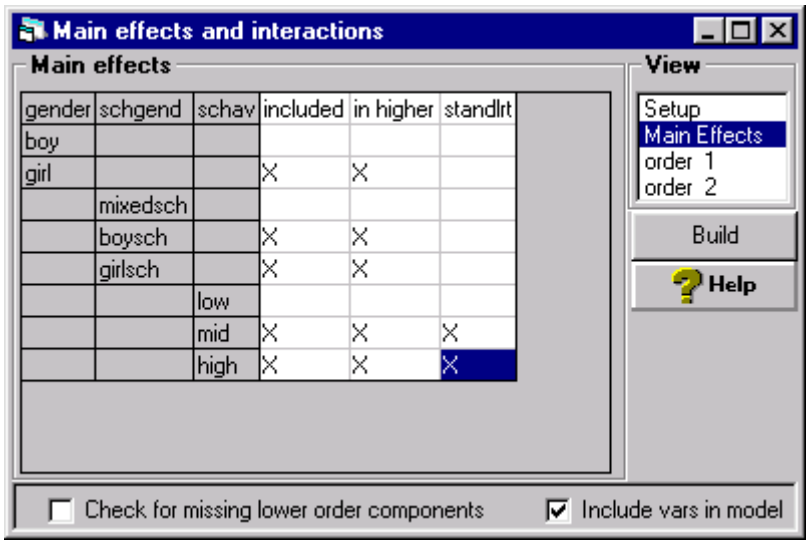
Select the **Main Effects and Interactions** window

Select **Main Effects** from the **View** panel

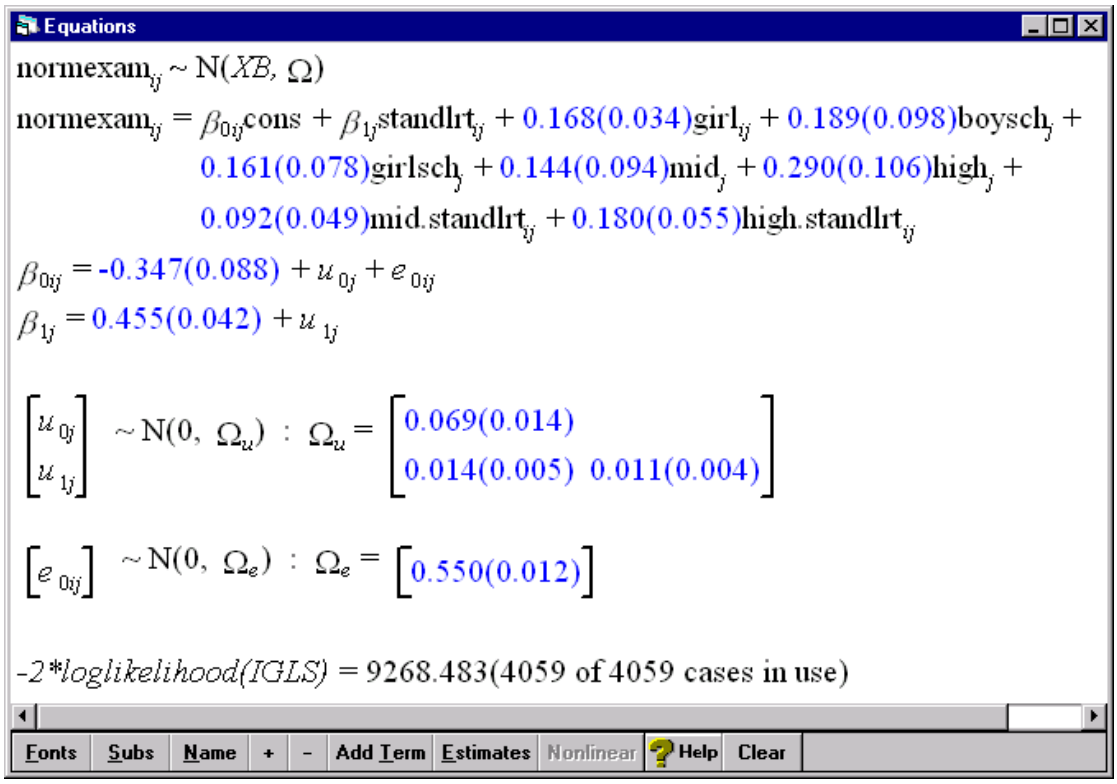
Click the **standlrt** column for the **mid** and **high** entries

Click **Build**

The **Main Effects and Interactions** window should look like this:



The model converges to :



The slope coefficient for **standlrt** for pupils from **low** intake ability schools is 0.455. For pupils from **mid** ability schools the slope is steeper 0.455+0.092 and for pupils from **high** ability schools the slope is steeper still 0.455+0.18. These two interaction terms have explained variability in the slope of **standlrt** in terms of a school level variable therefore the between school variability of the **standlrt** slope has been substantially reduced (from 0.015 to 0.011). Note that the previous contextual effects **boysch, girlsch, mid** and **high** all modified the intercept and therefore fitting these school level variables reduced the between school variability of the intercept.

We now have three different linear relationships between the output score(**normexam**) and the intake score(**standlrt**) for pupils from **low, mid** and **high** ability schools. The prediction line for **low** ability schools is

$$\hat{\beta}_0\text{cons} + \hat{\beta}_1\text{standlrt}_{ij}$$

The prediction line for the **high** ability schools is

$$\hat{\beta}_0\text{cons} + \hat{\beta}_1\text{standlrt}_{ij} + \hat{\beta}_6\text{high}_j + \hat{\beta}_8\text{high.standlrt}_{ij}$$

The difference between these two lines, that is the effect of being in a **high** ability school is

$\hat{\beta}_6 \text{high}_j + \hat{\beta}_8 \text{high.standlrt}_{ij}$

We can create this prediction function by

select the **Model** window

Select **Predictions**

Clear any existing prediction by clicking on **variable**

Select **Remove all explanatory variables** from the menu that appears

Click in turn on β_6, β_8

In the **output from prediction to** list select **c30**

Press Ctl-N and rename **C30** to **predab**

Click **Calc**

We can plot this function as follows :

Select the **Customised Graph** window

Select display number 5 **D5**

In the **y** list select **predab**

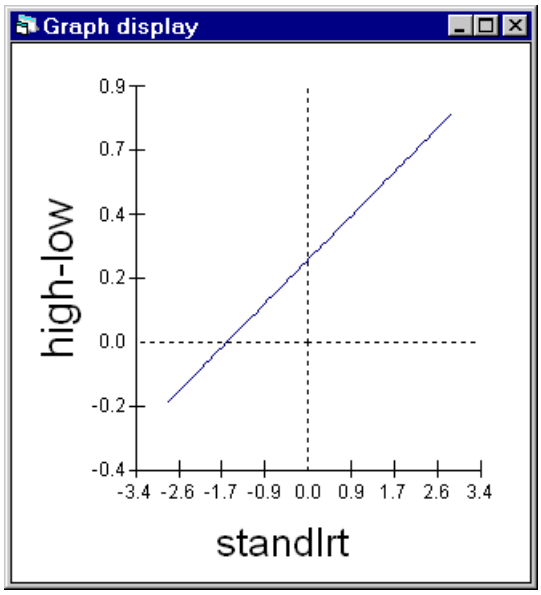
In the **x** list select **standlrt**

In the **plot type** list select **line**

In the **filter** list select **high**

Click **Apply**

Which produces



This graph shows how the effect of pupils being in a **high** ability school changes across the intake spectrum. On average very able pupils being educated in a **high** ability school score 0.9 of a standard deviation higher in their outcome score than they would if they were educated in a **low** ability school. Once a pupils' intake score drops below –1.7 then they fare progressively better in a **low** ability school. This finding has some educational interest but we do not pursue that here. We can put a 95% confidence band around this line by

Select the **Predictions** window

Edit the multiplier **S.E. of** to 1.96

In the **S.E. of** list select **Fixed**

In the corresponding **output to** list select **c31**

Click **Calc**

Select the **customised graph** window

Select **error bars** tab

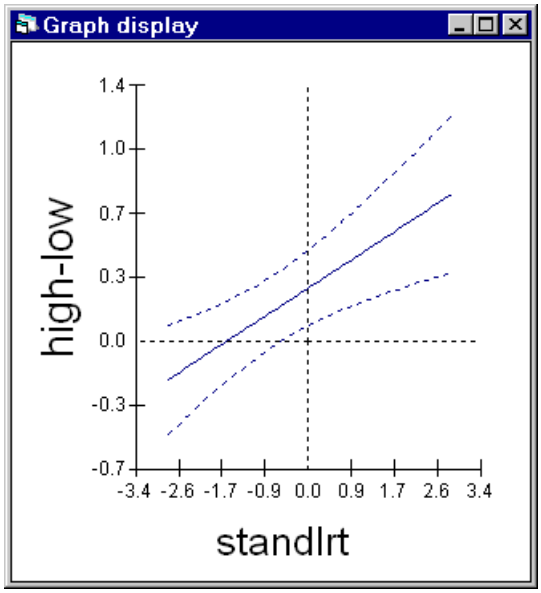
In the **y** errors + list select **c31**

In the **y** errors – list select c31

In the **y** error type list select **lines**

Click **Apply**

Which produces



Save your worksheet. We will be using it in the next chapter.

What you should have learnt from this chapter

- What is meant by contextual effects
- How to set up multilevel models with interaction terms

Chapter 6: Modelling the variance as a function of explanatory variables

From the fanning out pattern of the school summary lines seen in chapters 3 and 4 we know that schools are more variable for students with higher levels of **standlrt**. Another way of saying this is that the between school variance is a function of **standlrt**.

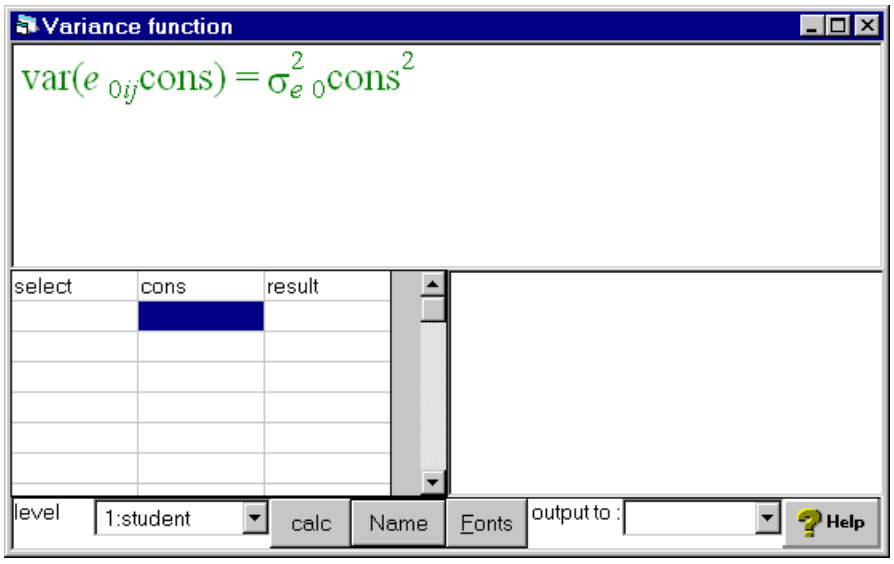
In *MLwiN* we always specify the random variation in terms of coefficients of explanatory variables, the total variance at each level is thus a function of these explanatory variables. These functions are displayed in the **Variance function** window.

Retrieve the worksheet from the end of chapter 5

Select the **Model** menu

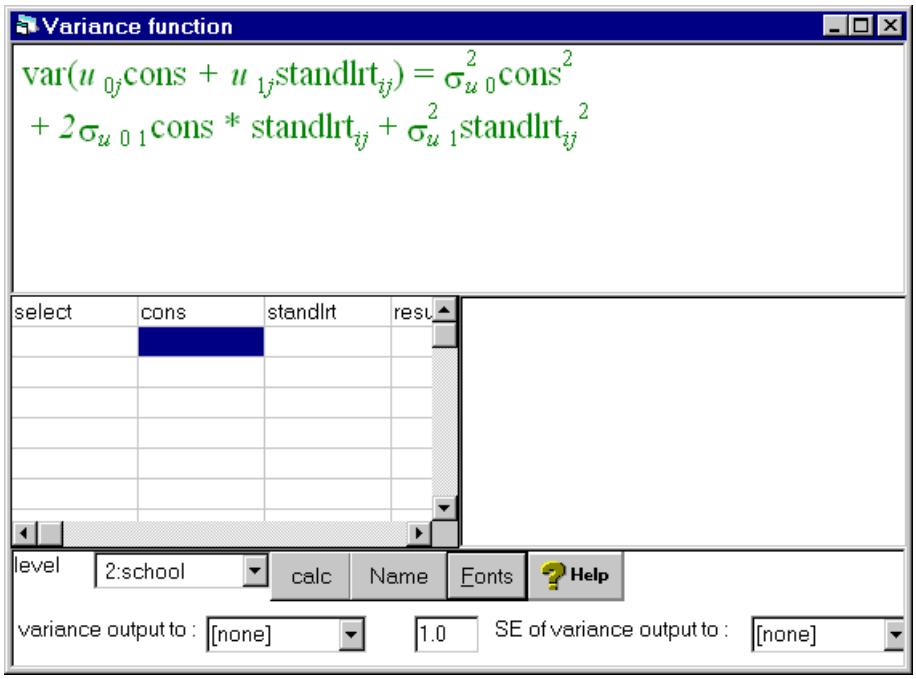
Select **Variance Function**

Click **Name** button in the V**ariance function** window



The initial display in this window is of the level 1 variance. In the present model we have simple (constant) variation at level 1, as the above equation shows. Now

In the **level** drop-down list, select **2:school**



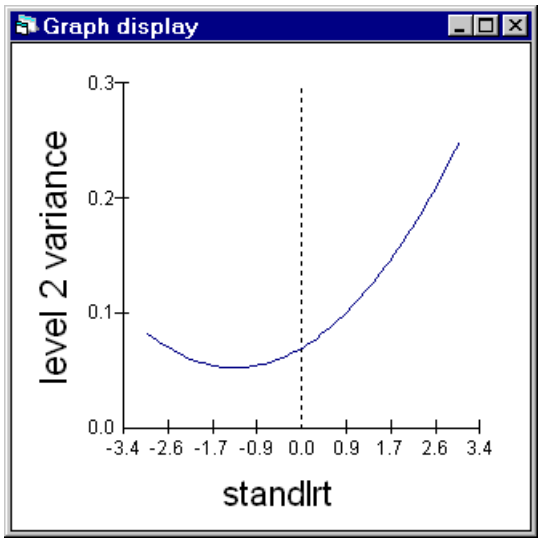
The function shown is simply the variance of the sum of two random coefficients times their respective explanatory variables, u_{0j}**cons** and u_{0j}**standlrt**$_{ij}$, written out explicitly. Given that **cons** is a vector of ones we see that the between school variance is a quadratic function of **standlrt** with coefficients formed by the set of level 2 random parameters. The intercept in the quadratic function is σ^2_{u0}, the linear term is $2\sigma_{u01}$ and the quadratic term is σ^2_{u1}. We can compute this function and the Variance function window provides us with a simple means of doing this.

The column in the window headed **select, cons, standlrt and result** are for computing individual values of the variance function. Since **standlrt** is a continuous variable it will be useful to calculate the level 2 variance for every value of **standlrt** that occurs**.**

In the **variance output to** list on the tool bar, select c30

Click **Calc**

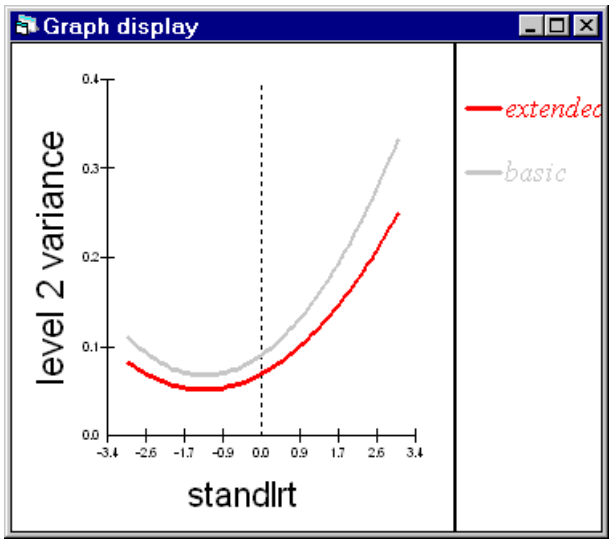
Now you can use the **customised graph** window to plot c30 against **standlrt:**



The above graph has had the y-axis rescaled to run between 0 and 0.3. The apparent pattern of greater variation between schools for students with extreme **standlrt** scores, especially high ones, is consistent with the plot of prediction lines for the schools we viewed earlier.

We need to be careful about over the interpretation of such plots. Polynomial functions are often unreliable at extremes of the data to which they are fitted. Another difficulty with using polynomals to model variances is that they may, for some values of the explanatory variables, predict a negative overall variance. To overcome this we can use nonlinear (negative exponential) functions to model variance. This is an advanced topic and for details see the *Advanced Modelling* Guide (Yang et al., 1999).

We can construct a similar level 2 variance plot for the basic random slope model, before extending the model by adding **gender, schgend** and **schav** explanatory variables. This can be illuminating because it shows us to what extent these variables are explaining between-school differences across the range of **standlrt**. This is left as an exercise for the reader but the graph comparing the between school variance for the two models is shown below.



We see in both models schools are more variable for students with high **standlrt** scores. The explanatory variables we added in the extended model explain about 25% of the between school variation across the spectrum of **standlrt.**

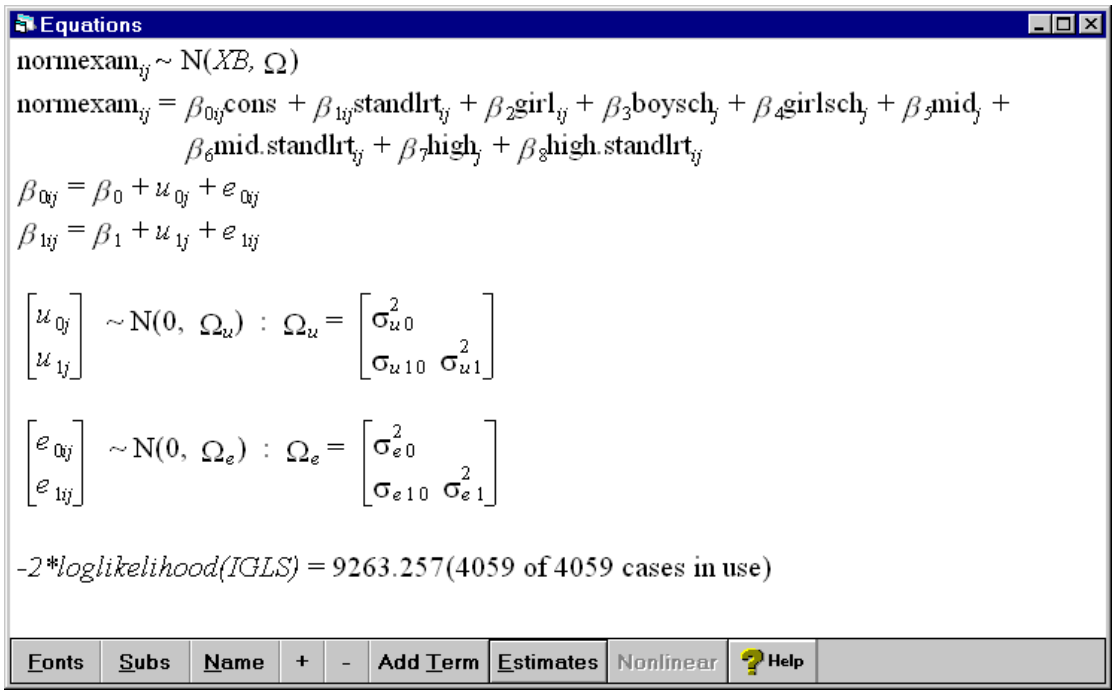
Complex variation at level 1

Until now we have assumed a constant variance at level 1. It may be that the student level departures around their school summary lines are not constant. They may change in magnitude at different levels of **standlrt** or be larger for boys than girls. In other words the student level variance may also be a function of explanatory variables.

Let's look and see if the pupil level variance changes as a function of **standlrt**. To do this we need to make the coefficient of **standlrt** random at the student level. To do this

In the equations window click on β_1 Check the box labelled **i(student)**

Which produces



Now β_1 the coefficient of **standlrt** has a school level random term u_{1j} and a student level random term e_{1ij} attached to it. As we have seen, at the school level we can think of the variance of the u_{1j} terms, that is σ^2_{u1} in two ways. Firstly, we can think of it as the between school variation in the slopes. Secondly we can think of it as a coefficient in a quadratic function that describes how the between school variation changes with respect to **standlrt**. Both conceptualisations are useful.

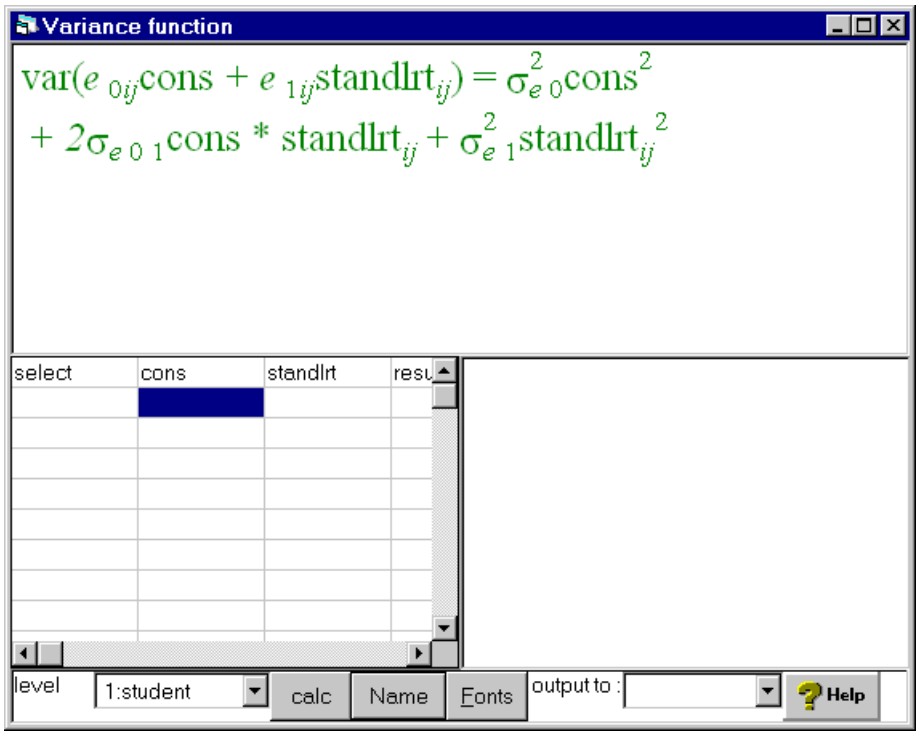
The situation at the student level is different. It does not make sense to think of the variance of the e_{1ij}'s, that is σ^2_{e1} as the between student variation in the slopes. This is because a student corresponds to only one data point and it is not possible to have a slope through one data point. However, the second conceptualisation where σ^2_{e1} is a coefficient in a function that describes how between student variation changes with respect to **standlrt** is both valid and useful**.** This means that in models with complex level 1 variation we do not think of the estimated random parameters as separate variances and covariances but rather as elements in a function that describes how the level 1 variation changes with respect to explanatory variables. The **variance function** window can be used to display the form of the function.

Run the model

Select the **variance function** menu

From the **level** drop down list select **1:student**

Which produces



As with level 2, we have a quadratic form for the level 1 variation. Let us evaluate the function for plotting

In the **output to** drop down list select **c31**

Click **calc**

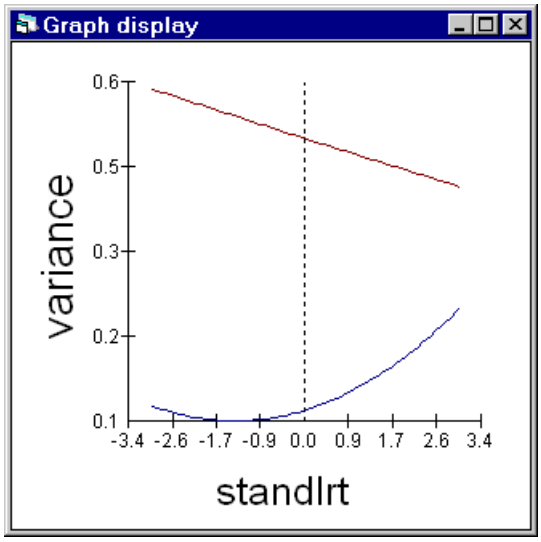
Now let's add the level 1 variance function to the graph containing the level 2 variance function.

Select the **customised graphs** window

Select the **display** used to plot the level 2 variance function

Add another data set with **y** as c31, **x** as **standlrt,** plotted as a red line

Which produces



The lower curved line is the between school variation. The higher straight line is the between student variation. If we look at the equations screen we can see that σ^2_{e1} is zero to 3 decimal places. The variance σ^2_{e1} acts as the quadratic coefficient in the level 1 variance function hence we have a straight line as the function is dominated by the other two terms. The general picture is that the between school variation increases as **standlrt** increases, whereas between student variation decreases with **standlrt.** This means the intra-school correlation(school variance / [school variance + student variance]) increases with **standlrt.** Therefore the effect of school is relatively greater for students with higher intake achievements.

Notice, as we pointed out earlier, that for high enough levels of **standlrt** the level 1 variance will be negative. In fact in the present data set such values of **standlrt** do not exist and the straight line is a reasonable approximation over the range of the data.

The student level variance functions are calculated from 4059 points, that is the 4059 students in the data set. The school level variance functions are calculated from only 65 points. This means that there is sufficient data at the student level to support estimation of more complex variance functions than at the school level.

Lets experiment by allowing the student level variance to be a function of gender as well as **standlrt**. We can also remove the σ^2_{e1} term which we have seen is negligible.

In the equations window click on β_2

Check the box labelled **i(student)**

The level 1 matrix Ω_e is now a 3 by 3 matrix.

Click on the σ^2_{e1} term.

You will be asked if you want to remove the term from the model. Click **yes**

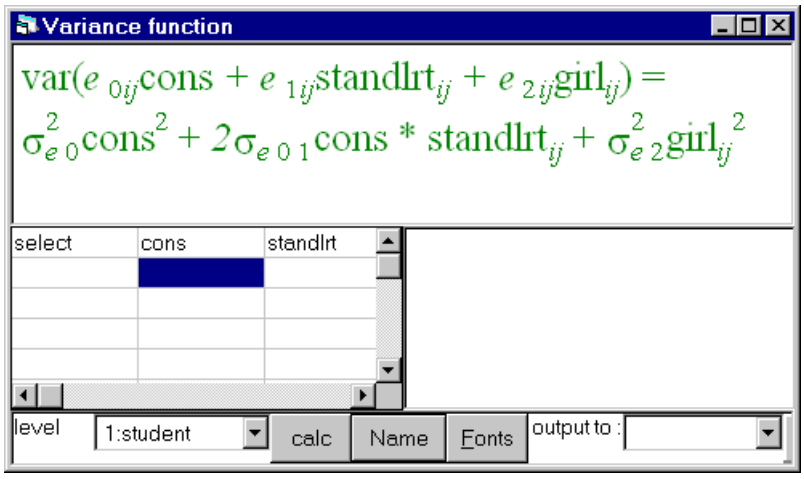
Do the same for σ_{e21} and σ_{e20}

When you remove terms from a covariance matrix in the **equations** window they are replaced with zeros. You can put back removed terms by clicking on the zeros.

Notice that the new level 1 parameter σ^2_{e2} is estimated as –0.054. You might be surprised at seeing a negative variance. However, remember at level 1 that the random parameters cannot be interpreted separately; instead they are elements in a function for the variance. What is important is that the function does not go negative within the range of the data.

[Note – *MLwiN* by default will allow negative values for individual variance parameters at level 1. However, at higher levels the default behaviour is to reset any negative variances and all associated covariances to zero. These defaults can be over-ridden in the **Estimation Control** window available by pressing the **Estimation Control** button on the main toolbar.]

Now use the variance function window to display what function is being fitted to the student level variance.



From the **equations** window we can see that $\{\sigma^2_{e0}, \sigma_{e01}, \sigma^2_{e2}\}$={0.583, -0.012, -.054}. Substituting these values into the function shown in the **variance function** window we get the student level variance for the boys is :

0.583 - 0.024 * **standlrt**

and for the girls is:

0.583 - 0.054 - 0.024***standlrt**

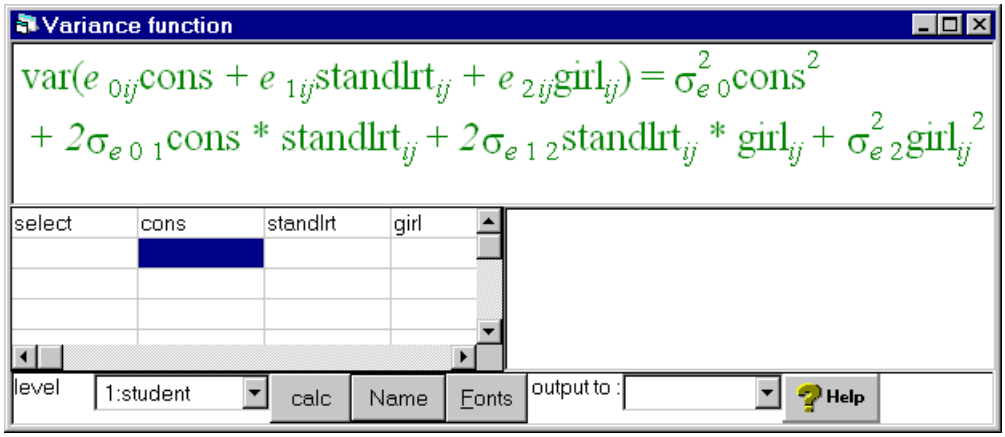
Note that we can get the mathematically equivalent result fitting the model with the following terms at level 1 : $\sigma^2_{e0}, \sigma_{e01}, \sigma_{e02}$. This is left as an exercise for the reader.

The line describing the between student variation for girls is lower than the boys line by 0.054. It could be that the lines have different slopes. We can see if this is the case by fitting a more complex model to the level 1 variance. In the **equations** window:

In the level 1 covariance matrix click on the right hand 0 on the bottom line. You will be asked if you want to add term **girl/standlrt.** Click **Yes.** Run the model

We obtain estimates for the level 1 parameters $\{\sigma^2_{e0}, \sigma_{e01}, \sigma_{e12}, \sigma^2_{e2}\}$={0.584, -0.032, 0.031,-0.058}

The updated variance function window now looks like this :



The level 1 variance for boys is now :

0.584 + 2 *(-0.032)***standlrt**=0.584 –0.064* **standlrt**

and for girls is:

0.584 +(2*(-0.032) +2*(0.031))* **standlrt**-0.058=0.526-0.02* **standlrt**

We can see the level 1 variance for girls is fairly constant across **standlrt.** For boys the level 1 variance function has a negative slope, indicating the boys who have high levels of **standlrt** are much less variable in their attainment**.** We can graph these functions :

In the **variance function** window set **output to:** list to c31

Press **calc**

Select the **customised graphs** window

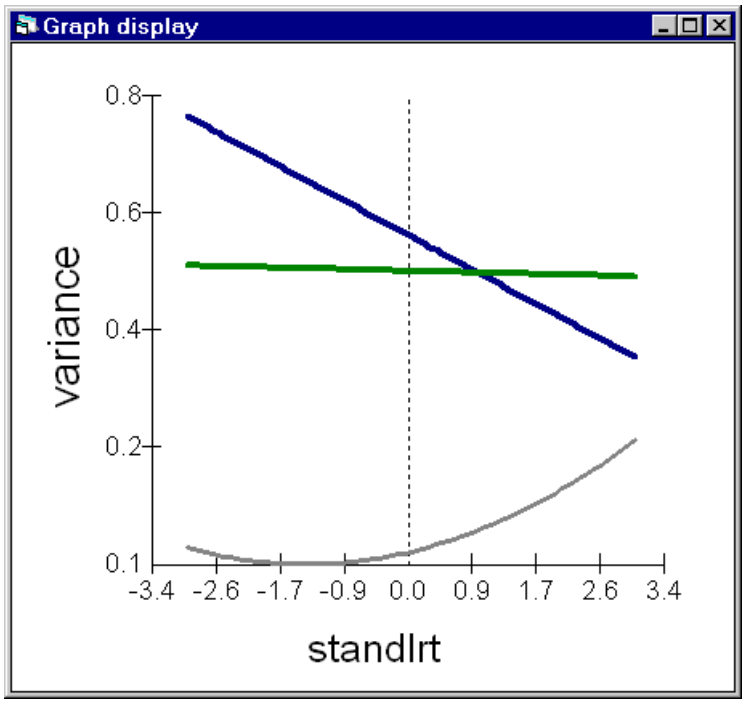
Select the **display** used to plot the level 2 variance function

Select the data set **y**=c31, **x=standlrt**

In the **group** list select **gender**

Click **Apply**

Which produces :



We see that the student level variance for boys drops from 0.8 to 0.4 across the spectrum of **standlrt**, whereas the student level variance for girls remains fairly constant at around 0.53.

We are now forming a general picture of the nature of the variability in our model at both the student and school levels of the hierarchy. The variability in schools' contributions to students progress is greater at extreme values of **standlrt,** particularly positive values. The variability in girls progress is fairly constant. However, the progress of low intake ability boys is very variable but this variability drops markedly as we move across the intake achievement range.

These complex patterns of variation give rise to intra-school correlations that change as a function of **standlrt** and **gender**. Modelling such intra-unit correlations that change as a function of explanatory variables provides a useful framework when addressing interesting substantive questions.

Fitting models which allow complex patterns of variation at level 1 can produce interesting substantive insights. Another advantage is that where there is very strong heterogeneity at level 1 failing to model it can lead to a serious model specification. In some cases the mis-specification can be so severe that the simpler model fails to converge but when the model is extended to allow for a complex level 1 variance structure convergence occurs. Usually the effects of the mis-specification are more subtle, you can find that failure to model complex level 1 variation can lead to inflated

estimates of higher level variances (that is between-student heterogeneity becomes incorporated in between-school variance parameters).

What you should have learnt from this chapter

That variance functions are a useful interpretation for viewing variability at the different levels in our model.

How to construct and graph variance functions in *MLwiN*

A more complex interpretation of intra unit correlations

Chapter 7: Getting started with your data and dealing with problems

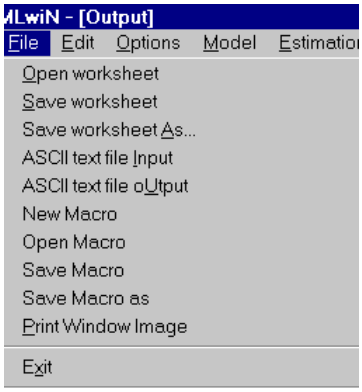
In the previous chapters we have used pre-prepared example data sets. This chapter describes how to get your data into *MLwiN*. We also give some advice on commonly experienced problems that occur once you start to fit models to your data.

Inputting your data set into *MLwiN*

MLwiN can only input and output numerical data. Data can be input and output to and from files or from the clipboard.

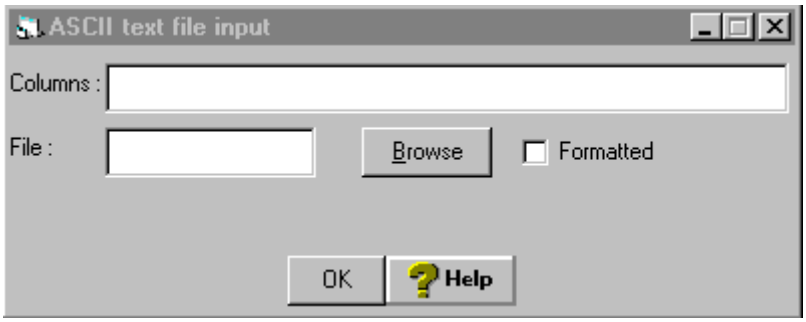
Inputting via files

When you click on the **File** menu you will obtain a drop down menu as follows



If you have data prepared in ASCII format you may use the **ASCII text file input** option from the above menu to input them.

Clicking on **ASCII text file Input/output** brings up the following window



To read in your data set do the following:

- In the **Columns** box type the column numbers or names into which the data are to be read. If columns are consecutive you can refer to them by typing the first and last separated by a '-', e.g. C2-C6.

- In the **File** box type either the full path for the data file or click **Browse** to display folder structure for a selection.

- If the data are space, comma or tab delimited, with the same number of variables per record, then clicking on **OK** will read the data into the specified columns.

If you wish to specify the input format, for example, in order to skip certain fields then check **Formatted**. This opens up a further box into which you type the data format. *MLwiN* recognises two formatting codes, *x* means read a variable of width *x* characters, *-x* means skip (ignore) *x* characters. Thus, to skip the first character in the file, input two 3-digit numbers, skip two characters, and input a 1 digit number, type the following in the box

-1,3,3,-2,1

Output of data to a specified text file operates in a similar way through the **ASCII text file output** option. If the data are not formatted the values are separated by spaces in the output file.

Common problems with ASCII data input

Some common problems that occur with ASCII text file inputting are

1. The data file includes a list of column names at the top; some packages save the column names to the top of the data file when using the save option.

2. The data file continues a character that is not a number, decimal point or minus sign, for example a letter.

3. The data file contained missing values that were converted to either blank spaces or illegal characters when the file was saved in another package.

4. The data file uses ',' rather than '.' to represent a decimal point. This is a problem for European users and although *MLwiN* will display worksheets using whichever representation is set on a computer, when inputting data from another package the '.' must be used.

5. The number of columns given in the ASCII text file input box does not correspond to the number of columns in the input file.

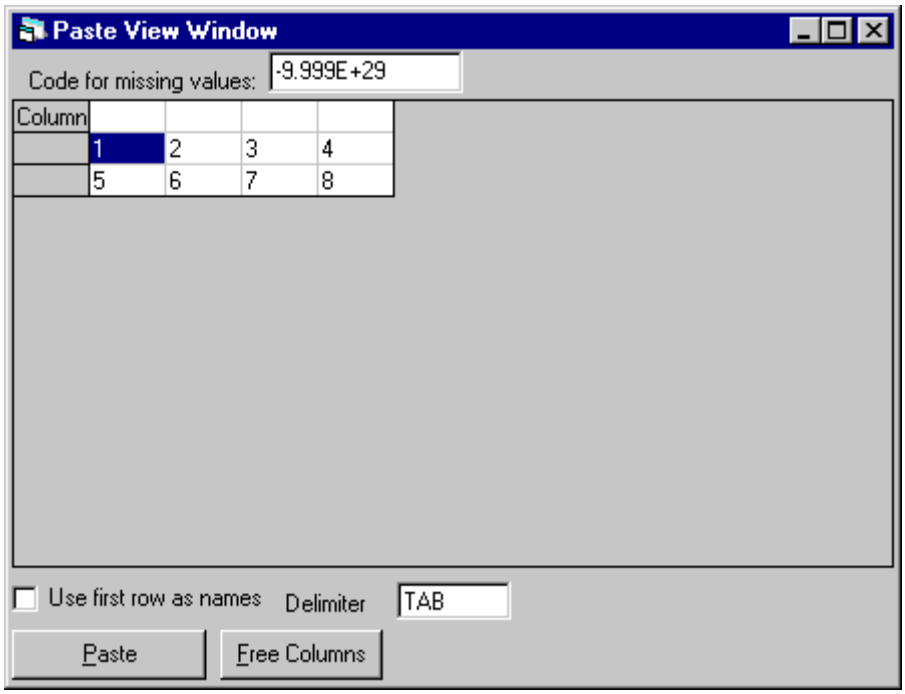
All of these reasons can be checked by viewing the data set using the software package that the data are being exported from, or by looking at the data file with a word processor. To correct the data most packages have a Find and Replace option and this can be used to globally convert any illegal characters. If the data contain missing values it is sensible to convert these to a unique value, for example –999 which can then be exploited after the data have been successfully input into *MLwiN*. We will say more about missing values later in this chapter.

If you have a very large data set make sure that you have specified a large enough worksheet size using the **Options/worksheet** menu. It is also a good idea to read in a subset of columns at a time (into consecutive sets of columns).

Pasting data from the clipboard

If you have your data in another package such as EXCEL or SPSS it may often be more convenient to copy your data from these packages and then paste them into *MLwiN*.

If you have copied data onto the clipboard from another application then they can be pasted into *MLwiN* through the **Paste** option on the **Edit** menu. For example, if we have a 2 by 4 table of numbers in the clipboard and we select **Paste**, the following screen appears



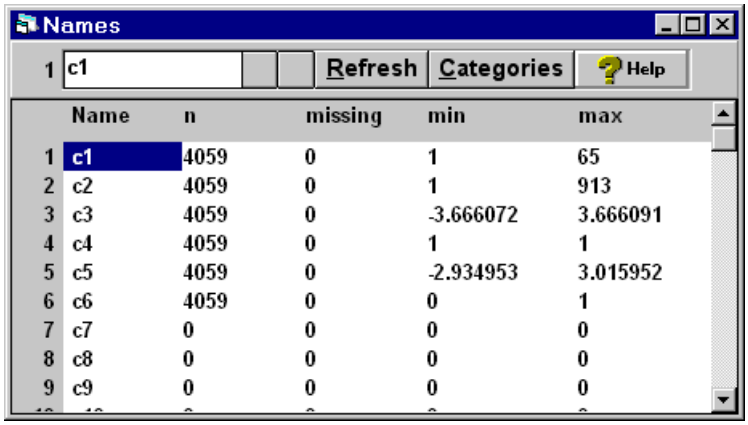
This window allows you to view the data and assign it to *MLwiN* columns. You can select the next free columns on the worksheet by pressing the **Free Columns** button. You can also choose which *MLwiN* columns the data are to be assigned to by clicking in the top row of the data table and selecting *MLwiN* columns from the list that appears.

If the first row of your pasted data contains variable names then checking the U**se first row as names** box will assign these names to the *MLwiN* copies of the variables.

As with ASCII input, if you have a very large data set make sure that you have specified a large enough worksheet size using the **Options/worksheet** menu. It is also a good idea to paste in a subset of columns at a time (into consecutive sets of columns).

Naming Columns

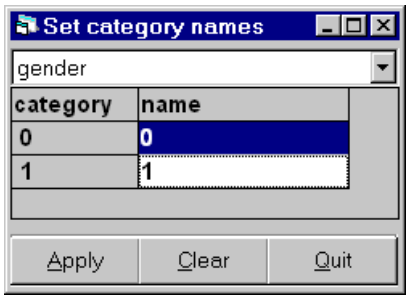
Apart from the paste window, *MLwiN* does not allow the user to input the column names from a file into *MLwiN*. Columns can instead be named in the **Names** window which is available from the **Data Manipulation** menu. An example Names window is shown below



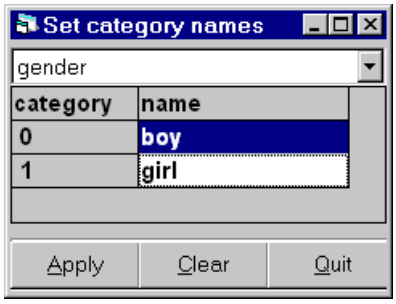
A column is selected by clicking on it's column number (c1, c2, c3 etc) in the column headed **Name** in the table. The selected column name is also displayed in the text box on the toolbar where it can be edited to specify any desired name. After editing pressing **return** updates the column name.

Category Names

When data is categorical, names can also be given to the individual categories. This is useful as it allows *MLwiN* to create and name dummy variables for analysis, and to annotate tables etc. Column 6 in the above data set is gender. We can name the categories by selecting column 6 in the **Names** window and then pressing the **Categories** button. This produces the following screen



Clicking in the name column and typing text allows category names to be assigned, for example



Missing Data

MLwiN uses a single value coding for missing data. The default value is a large negative number (-9.999E+29) - the *system missing* value. The numerical value to represent missing data can be set by the user on the **Numbers** screen available from the **Options** menu.

You may have inputted data containing a specific missing value code, –99, say. In this case you should set *MLwiN*'s missing data value to –99.

What happens when a user specifies a missing value code is that the code (say -99) is changed to that of the above system missing value. There is a 1-step recovery from this: if you change the missing code again you will be prompted to see whether you wish the previous code to revert to its original value. If not then the old *and* new codes are treated as missing. Note that you can also use the recode facility to change a missing value to another code (see HELP topic). The calculate window allows the missing code to be used in logical expressions.

It is important to note that missing data are automatically ignored in calculations and, for example, that a likelihood ratio statistic comparing two models with different amounts of missing data is not valid. The **Equations** window reports how many cases were used in each model.

Level Identification Columns

These variables often cause problems for two reasons. First these columns are simply indicators of the various levels in the data and will often contain letters as well as numbers. Secondly these variables are sometimes given very large values, for example a level 2 unit number 1234567890. This should be avoided as due to data precision this level 2 unit will be indistinguishable from level 2 unit 1234567891 as both are stored as 1.2345678E+09. To solve the first of these problems it may be possible to convert some letters into numbers, for example A = 1, B = 2 etc but this depends on the exact nature of the variable. For the second problem it may be possible to subtract a large number, for example 1234567880 from all level 2 unit identifications to give more sensible codes.

Saving the worksheet

Once you have input and named your data you should save your data as an *MLwiN* worksheet using the **Save worksheet** option on the **File** menu. While working with *MLwiN* it is well worth saving your worksheet at regular intervals, this is essentially backing up your work.

Fitting Models in *MLwiN*

Once you have input the data into *MLwiN*, named the columns and saved the worksheet many it is often tempting to go straight ahead and fit a really complicated model with lots of fixed and random effects. Then you may well come across several problems, for example the model does not converge, has numerical problems or gives unexpected answers. The main piece of advice here is that multilevel modelling is like any other type of statistical modelling and a useful strategy is to start by fitting simple models and slowly increase the complexity. In the rest of this section we will list some of the main pointers that should be followed to reduce frustration while trying to fit multilevel models in *MLwiN*.

What are you trying to model?

It is important before starting to fit models to your dataset to know as much as possible about your data and to establish what questions you are trying to answer. It is important to identify which variable(s) are your response variable(s) of interest. It is also important to establish, particularly if you are new to multilevel modelling, what is meant by the terms: levels, predictors, fixed effects and random effects, and to identify which variables in your dataset represent level indicators and which are measured variables. If you are not sure of these terms mean then you need to work through chapters 1 to 6 of this manual before proceeding with your own data.

Do you really need to fit a multilevel model?

It is always a good idea to do some more basic statistical analysis before proceeding on to multilevel modelling. Plotting the response variable against several predictors will allow you to visually see if there are any strong relationships. Fitting simple single level models before proceeding onto multilevel models is also a good idea, particularly as the fixed effects estimates from a single level model should generally be similar to those achieved by an equivalent multilevel model.

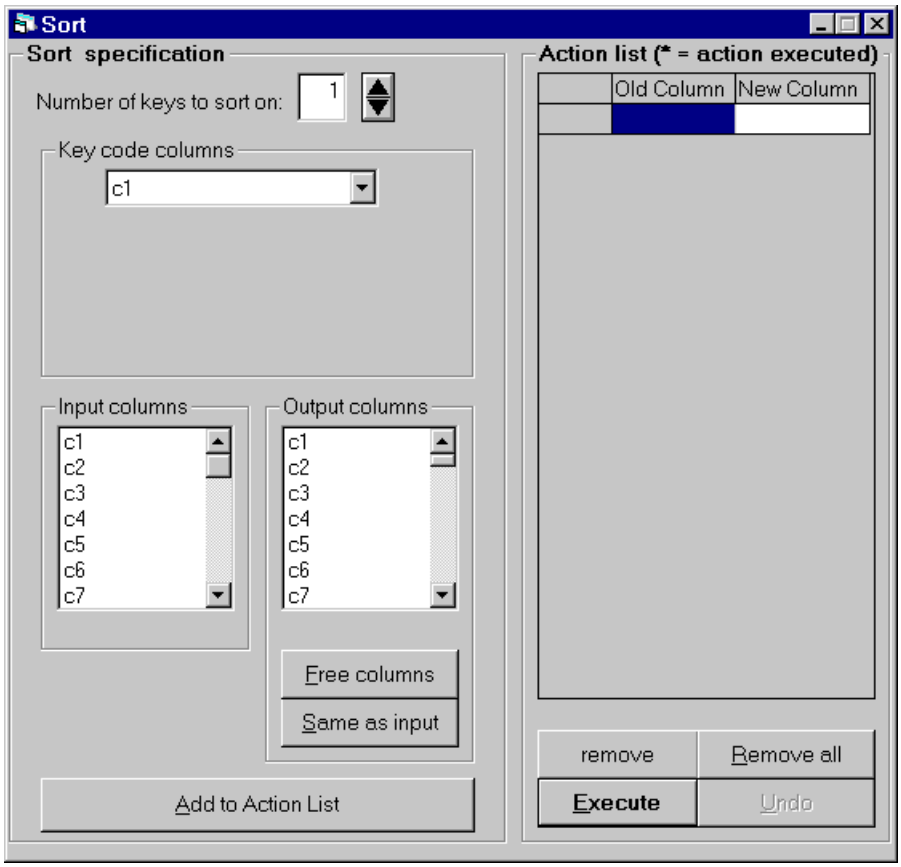
One point to note is that just because a model has more levels, more fixed effects and more random effects this does not automatically mean that it will be a better model and often the opposite is true. A distinction should be made here between trying to fit a multilevel model to a dataset that is too small and to a dataset where there is no higher level variation. A dataset that only has 4 level 2 units is best fitted as a single level model with the level 2 units included as 3 dummy variables. Fitting a multilevel model to this data will almost certainly report no level 2 variation. However, this is not a generalisable statement, we simply have not sampled enough level 2 units.

Have you sorted your dataset?

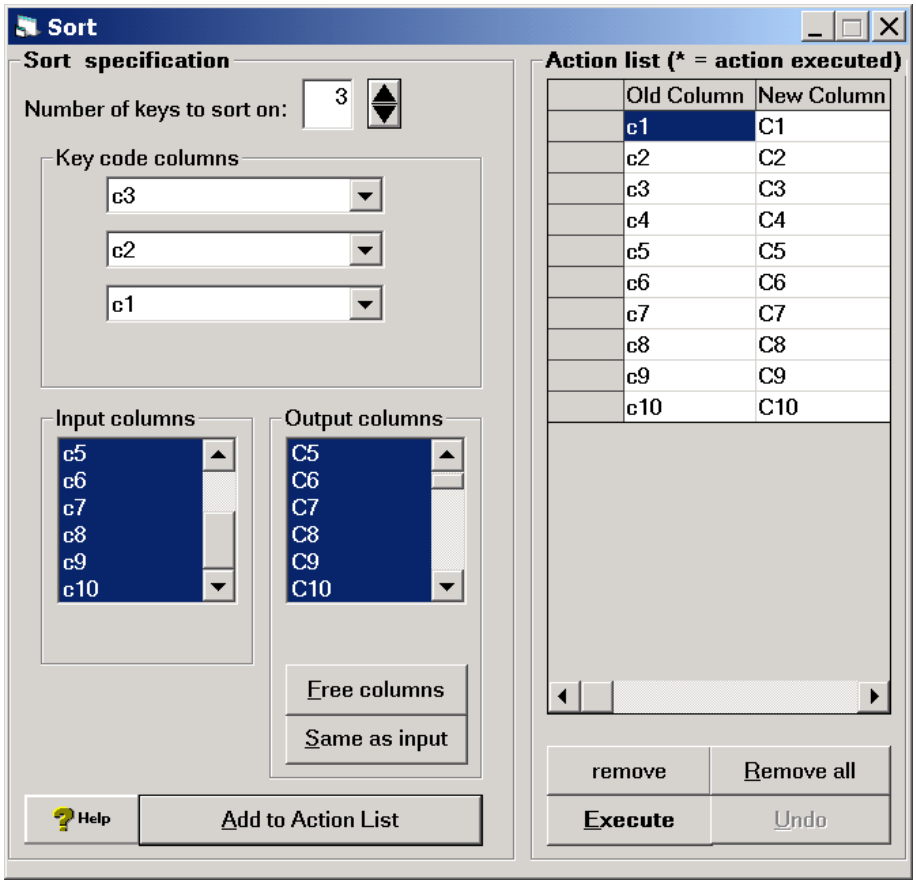
The most common mistake users make when trying to fit a multilevel model to their dataset is that they do not sort the data into the correct format. This is an easy mistake to make as all the examples in this manual have already been sorted into the correct format. The data must be sorted so that all records for the same highest level unit are grouped together and within this group of records all records for a particular lower level unit are grouped together. For example the following represents the first few records of a sorted 3 level model:

Level 3 ID	Level 2 ID	Level 1 ID
1	1	1
1	1	2
1	1	3
1	2	1
1	2	2
1	2	3
2	1	1
2	1	2

The **Sort** option on the **Data Manipulation** menu can be used to sort data. This option brings up the following screen:



We assume that the level 1, 2 and 3 identifiers of our model are stored in columns C1, C2 and C3 respectively and that the response and all predictor variables can be found in columns C4-C10. Then to sort this data structure correctly we need to set the following on the **Sort** window as shown in the window below.



A full explanation of how to use both the sort window and other data manipulation windows can be found in the on-line Help system. Note that all of the columns C1-C10 must be of the same length and contain data for the **sort** to work correctly. Many users ask why the software cannot sort the data itself and there are several reasons for this. Firstly the software doesn't know which columns the user wants to sort (C1-C10 in the above example). Secondly the software cannot currently simply take the unsorted columns of data from the worksheet that are used for a particular model and perform the sorting itself. One reason for this can be seen if we consider the above table of sorted data. If instead of a 3 level model we wanted to drop level 3, we would then have several records that have the same level 2 ID (1), but do not actually come from the same level 2 unit. In an educational scenario they could be from class 1 in school 1 and from class 1 in school 2 which are clearly not the same unit.

Once you have sorted your data and set up a model always check the **hierarchy viewer** on the **Model** menu and ensure that the data structure the software reports (in terms of number of units at each level) is as you expect.

Have you built up your model from a variance components model?

A sensible way of fitting a multilevel model is to start with the basic variance components model (as in the tutorial example). Then you can build up models of increasing complexity by adding predictors that are deemed to be important and checking if they have substantial and/or significant fixed or random coefficients. If instead you add lots of predictors into your model and have convergence problems it may be difficult to establish which predictor is causing the problem. Building up a model by adding variables one at a time and using the NEXT option rather than START also has less chance of convergence problems with the IGLS and RIGLS methods. This is because the estimates from the last model fitted are used as starting values for the new model.

Have you centered your predictor variables?

If you have continuous predictor variables there are many benefits to be gained from centering these values (subtracting their means). The first benefit is that often the intercept term in the model is easier to interpret as it is now the predicted value for a subject that has average values for each explanatory variable. This is generally more useful than the response value for a subject with zero for all predictors, as zero may not be a typical value for the corresponding explanatory variable. Centering predictor variables can also reduce the chances of numerical errors in the IGLS and RIGLS estimation methods and reduce the correlation in the chains produced by the MCMC methods.

What you should have learnt from this chapter

- How to input data into *MLwiN*.
- How to sort data and set missing values.
- How to set up categorical variables.
- An awareness of some of the common mistakes users can make when modelling data.

PART 2. MORE COMPLEX MODELS

Chapter 8: Modelling binary and general Binomial responses

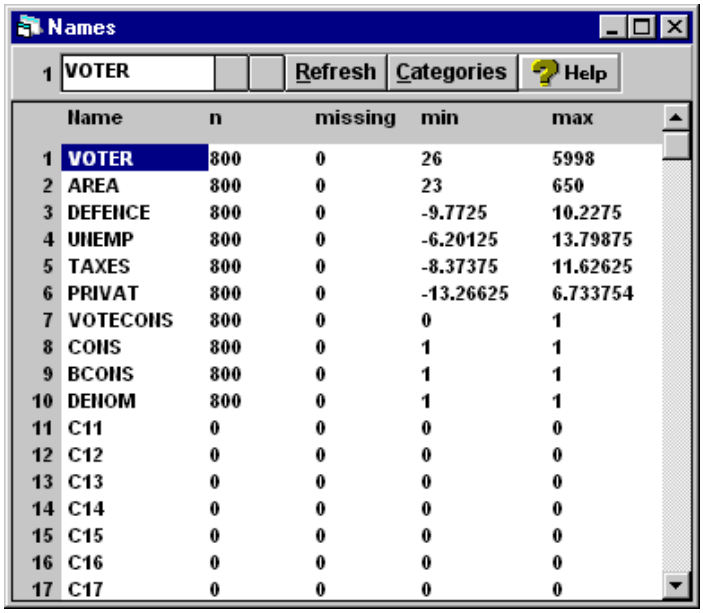
Binary response models

In this chapter we will look at how to fit models to data where the response variable is measured as a simple yes/no for each individual in a study. These modelling procedures will then be applied to the more general case of data where the response is a proportion. In *MLwiN* the analysis of such responses and also responses that are counts, which will be considered in the next chapter, is specified in the equations window by bringing up the required error distribution. For more complex models with discrete responses, such as multicategory data (ordered and unordered) and event history data, special-purpose macros have been developed and their use is described in an advanced user's guide (Yang et al., 1999). Further details are also provided in the *MLwiN* help system.

We begin with a description of the data set to be used, followed by the specification of some models and then some analysis.

A dataset on political voting

To begin this example we will open the data set bes83.ws by using the **Open Worksheet** option from the **File** menu. Then if we open the **Names** window from the **Data Manipulation** menu we will see the following.



Names

1 VOTER | Refresh | Categories | Help

	Name	n	missing	min	max
1	VOTER	800	0	26	5998
2	AREA	800	0	23	650
3	DEFENCE	800	0	-9.7725	10.2275
4	UNEMP	800	0	-6.20125	13.79875
5	TAXES	800	0	-8.37375	11.62625
6	PRIVAT	800	0	-13.26625	6.733754
7	VOTECONS	800	0	0	1
8	CONS	800	0	1	1
9	BCONS	800	0	1	1
10	DENOM	800	0	1	1
11	C11	0	0	0	0
12	C12	0	0	0	0
13	C13	0	0	0	0
14	C14	0	0	0	0
15	C15	0	0	0	0
16	C16	0	0	0	0
17	C17	0	0	0	0

These data come from the longitudinal component of the British Election Study (see Heath et al. 1996). It consists of a subsample of 800 voters grouped within 110 voting constituencies who were asked how they voted in the 1983 British general election. For our purposes the responses are classified simply as to whether or not they voted Conservative. The variables are defined as follows:

Voter is the identification of each individual voter.

Area identifies the constituency they voted in.

Defence is a 21 point scale of attitudes towards nuclear weapons, with low scores indicating respondents were against Britain possessing them. The variable is centred about its mean value.

Unemp is a 21 point scale of attitudes towards unemployment, with low scores indicating strong opposition and high scores a preference for more unemployment if it leads to lower inflation. The variable is centred about its mean value.

Taxes is a 21 point scale of attitudes towards tax cuts, with low scores indicating a preference for higher taxes to pay for more government spending. The variable is centred around its mean value.

Privat is a 21 point scale of attitudes towards privatisation of public services, with low scores indicating opposition. The variable is centred around its mean value.

Votecons is 1 if the respondent voted Conservative and 0 otherwise.

Our main interest lies in seeing how the combination of these explanatory variables influences the probability of voting Conservative and the extent of between-constituency variation. We will first however fit a simple model with no predictor variables.

Specifying a simple binary response model

We have a basic binary response (whether or not the respondent voted or intended to vote Conservative) and we will firstly fit a variance components model to assess the average probability of voting Conservative along with the between-constituency variation before adjusting for the voters' attitudes.

Let π_{ij} be the probability that the *i*th respondent in the *j*th constituency voted Conservative. We define this probability as a function of the intercept as follows:

$$\text{logit}(\pi_{ij}) = \beta_{1j}x_1, \text{ where } \beta_{1j} = \beta_1 + u_{1j}$$

In the above equation x_1 (with no further subscript) takes the value 1 for all respondents. Its coefficient β_{1j} indicates that we are modelling the intercept in this relationship as random at the level of the constituency. We have used x_1 for this purpose since we shall need to use x_0 to specify the variation at level 1. We have used the logit link function here. *MLwiN* also allows a probit or a complementary log-log function (see McCullagh and Nelder, 1989, for further discussion).

The full model can now be written as

$$y_{ij} = \pi_{ij} + e_{0ij}\ x_0$$

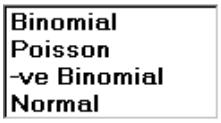
where the y_{ij} are the observed (0,1) responses, x_0 is equal to 1, and we need to define the variance of the level 1 random terms e_{0ij}. The standard assumption is that the response y_{ij} is distributed as Binomial $(1, \pi_{ij})$. We may write this distributional assumption in a general form as:

$$y_{ij} \sim \text{Binomial}(n_{ij}, \pi_{ij})$$

where in our case the n_{ij} are all equal to 1, so that the variance of e_{0ij} is $\pi_{ij}(1-\pi_{ij})$. We shall see later how this is utilised.

The general distributional form is used to model proportions (see later), where each proportion y_{ij} is based on n_{ij} observations and has a denominator n_{ij} so that the variance of e_{0ij} will be π_{ij} $(1-\pi_{ij})/n_{ij}$. The present example is a special case in which this denominator is everywhere 1, but *MLwiN* nevertheless requires a column of denominators to have been defined. The standard name **denom** is used for this column, and you will see in the **Names** window that such a column has already been set up in the worksheet. We now show how to specify this model in *MLwiN*.

Open the **Equations** window from the **Models** menu and click on the upper case N which by default indicates a Normal response distribution. You will see the following drop down menu appear:



Click on **Binomial** and the display will change to the following

$$y \sim \text{Binomial}(n, \pi)$$
$$y = \pi$$
$$\text{logit}(\pi) = \beta_0 x_0$$

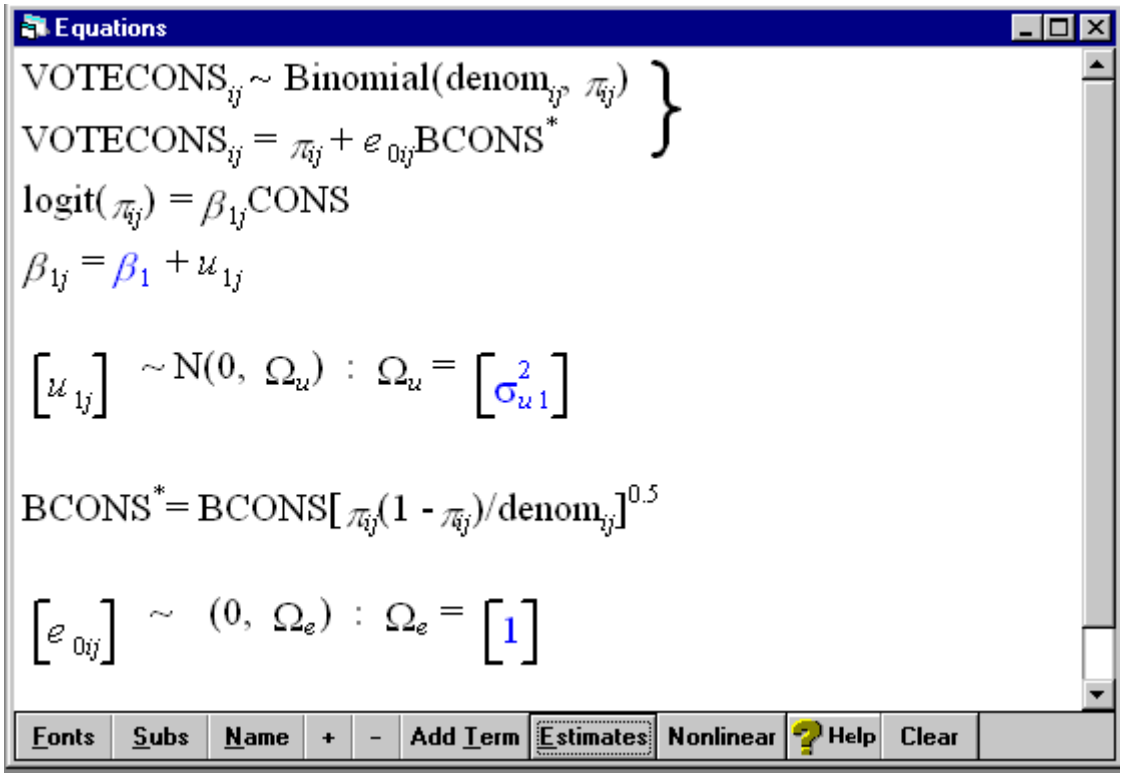
This can now be elaborated by filling in the response variable, the levels and the explanatory variables as in the standard Normal case. First of all we should define the response variable, **votecons**, by clicking on *y*, and then from the same dialogue box select a 2-level model with **area** at level 2, and **voter** at level 1. We now want to add the level 1 binomial variation to the model. To do this we require a new variable, equal to 1, but distinct from the existing constant variable, **cons**. This has already been defined and named **bcons** in column 9 of the worksheet.

Now click on x_0 and define '**bcons**' as having a random coefficient at level 1 (voter) only and not being in the fixed part. Click on **done** and the following screen will appear:

$$y_{ij} \sim \text{Binomial}(n_{ij}, \pi_{ij})$$
$$y_{ij} = \pi_{ij} + e_{0ij} x^*_0$$
$$\text{logit}(\pi_{ij}) =$$

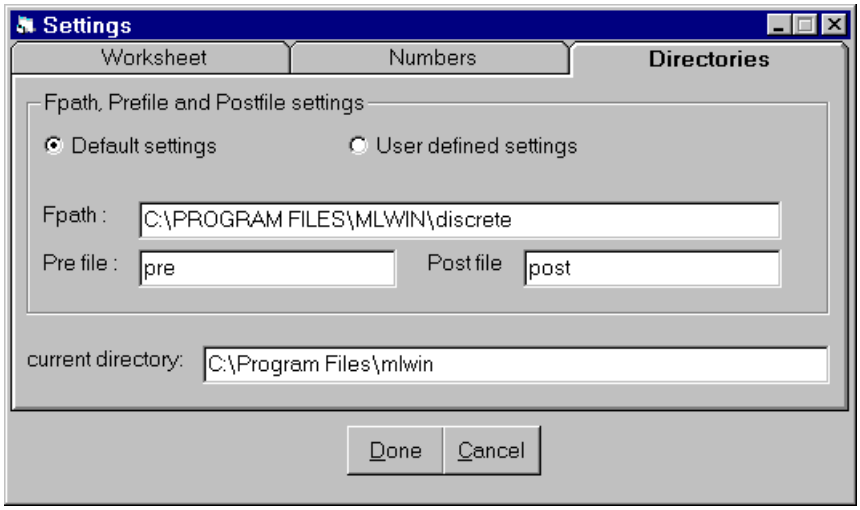
Note that the default **logit** link function appears; this can be changed to the **probit** or complementary **log-log** link functions by clicking on it. Note also that the variable x_0^* has appeared on the second line; this is because we defined it as having only a level 1 random coefficient with no contribution to the fixed part of the model. It is related to our original x_0 according to the formula on line 6 of the following window. Now we can fill in the intercept term into the model. Click on the **Add Term** button and then click on the red x_1 that appears. Choose **cons** from the list and select this variable as a fixed effect and with a random coefficient at level 2 (Area).

Once this is done we can display the first model, with names, as follows



Note that at level 1 a random coefficient e_{0ij} is defined with a zero mean and a variance which is constrained to equal 1. This implies that we have binomial variation since now from line 2 of the above window we see the variance is σ_e^2 $(\text{BCONS}^*)^2 = \pi(1-\pi)/\text{denom}$. This constraint is the default assumption and can be altered via the **Nonlinear estimation** window which is described in the next section. The asterisk (*) appears by the level 1 explanatory variable to indicate that the methods used to carry out the estimation will transform **bcons** to have the form given. The basic *MLwiN* estimation procedure uses a quasilikelihood estimation procedure (see below). We shall also look at alternative estimation procedures (in later chapters) which nevertheless use the quasilikelihood estimates as a starting point. Note again that because **bcons** is used to define level 1, the remaining explanatory variables start their numbering at 1 rather than zero.

Finally, to run this model and other discrete response models, you must specify the path name for a set of macros which carries out the necessary manipulations. The Help system describes the use of these in more detail. Clicking on the **directories** item on the **Options** menu brings up the following screen:

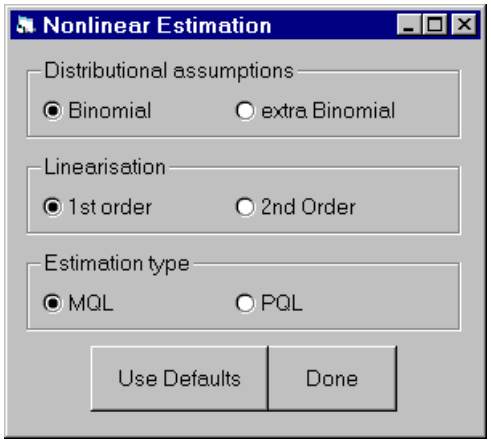


Under **Fpath** is the default location where the *MLwiN* setup will have placed these files. Click on **Done** to confirm that these are there (unless you have moved them elsewhere).

It is a good idea at this stage to save the worksheet, using a different name.

Quasilikelihood – MQL/PQL methods

The default estimation procedure is to use the first order marginal quasilikelihood (MQL) method. For a detailed description of the use of these procedures see Goldstein and Rasbash (1996) or Goldstein (1995a, Chapter 7). Clicking on **nonlinear** at the bottom of the **equations** window brings up the following default estimation screen:



In the present case we shall start using first order MQL estimation, then switch to second order penalised quasilikelihood (PQL) estimation. The former is somewhat faster, but tends to underestimate variance parameters when there are few level 1 units per level 2 unit or where higher level variances are large (see Goldstein and Rasbash, 1996, for a detailed discussion). The second order PQL estimates are the least biased, but in some circumstances any of the combinations other than first order MQL may fail to converge. After clicking on **Done**, click on **Estimation control** and under **IGLS/RIGLS** select **RIGLS**.

It is a good idea to change the default convergence tolerance criterion from 2 to 3 in this model as all the estimates are small. This convergence criterion means that the relative change from one iteration to the next has to be less than 10^{-3} for all parameter values in order that convergence is achieved.

Click on **Start** and convergence is achieved after iteration 4 as follows:

$$\left.\begin{aligned} y_{ij} &\sim \text{Binomial}(n_{ij}, \pi_{ij}) \\ y_{ij} &= \pi_{ij} + e_{0ij}x^{*}_{0} \end{aligned}\right\}$$

$$\text{logit}(\pi_{ij}) = \beta_{1j}x_{1}$$

$$\beta_{1j} = -0.248(0.081) + u_{1j}$$

$$\begin{bmatrix} u_{1j} \end{bmatrix} \sim \text{N}(0, \Omega_u) : \Omega_u = \begin{bmatrix} 0.134(0.091) \end{bmatrix}$$

$$x^{*}_{0} = x_{0}[\pi_{ij}(1 - \pi_{ij})/n_{ij}]^{0.5}$$

$$\begin{bmatrix} e_{0ij} \end{bmatrix} \sim (0, \Omega_e) : \Omega_e = \begin{bmatrix} 1.000(0.000) \end{bmatrix}$$

In this model we see that the intercept term is much larger than its standard error. From this value we can estimate that the median proportion of people voting Conservative is

$[1+\exp(0.248)]^{-1} = 0.438$

Note that because the link function is non-linear this is not exactly the overall mean – see Goldstein (1995a) for a further discussion. The level 2 variance is quite small and only slightly larger than its standard error. We can construct very approximate confidence intervals and significance tests using this standard error, but we can obtain more accurate intervals using either the MCMC methods or the bootstrap as shown in later chapters.

Note also that the variance of the e_{0ij} is given as 1.000. This means that we are fitting a model that has a Binomial error distribution since the overall level 1 variance is then $(x^*_0)^2$ i.e. π_{ij} $(1-\pi_{ij})/n_{ij}$ since $x_0 = 1$. We shall discuss a relaxation of this assumption to allow 'extra-Binomial' variation for modelling general Binomial responses below.

As described earlier in this chapter we have several attitudinal predictors that we can also add to our model. Using the **Add term** button we can add all four predictor variables (defence, unemp, taxes and privat) as fixed effects into the model, and then press **Start** to run the model, again using first order MQL. Convergence is achieved after 5 iterations and the results are as follows:

$$\left.\begin{aligned} \text{votecons}_{ij} &\sim \text{Binomial}(\text{denom}_{ij}, \pi_{ij}) \\ \text{votecons}_{ij} &= \pi_{ij} + e_{0ij}\text{bcons}^* \end{aligned}\right\}$$

$$\text{logit}(\pi_{ij}) = \beta_{1j}\text{cons} + 0.089(0.018)\text{defence}_{ij} + 0.067(0.013)\text{unemp}_{ij} + 0.044(0.019)\text{taxes}_{ij} + 0.138(0.018)\text{privat}_{ij}$$

$$\beta_{1j} = -0.355(0.092) + u_{1j}$$

$$\begin{bmatrix} u_{1j} \end{bmatrix} \sim \text{N}(0,\ \Omega_u) \ :\ \Omega_u = \begin{bmatrix} 0.144(0.114) \end{bmatrix}$$

$$\text{bcons}^* = \text{bcons}[\pi_{ij}(1-\pi_{ij})/\text{denom}_{ij}]^{0.5}$$

$$\begin{bmatrix} e_{0ij} \end{bmatrix} \sim (0,\ \Omega_e) \ :\ \Omega_e = \begin{bmatrix} 1.000(0.000) \end{bmatrix}$$

Here we can see that all the fixed effects are much larger than their standard errors. We can now calculate the estimated median proportion voting Conservative who have average values for all the attitude variables (with the explanatory variables $x_2 - x_5$ set to zero as the predictors are all centred), which is

$[1+ \exp(0.355)]^{-1} = 0.412$.

Now press the **Nonlinear** button on the equation window and from the **Nonlinear estimation** window select 2nd order PQL and click on **More** on the toolbar. After a few more iterations we obtain the following estimates:

$$\left.\begin{aligned} \text{votecons}_{ij} &\sim \text{Binomial}(\text{denom}_{ij}, \pi_{ij}) \\ \text{votecons}_{ij} &= \pi_{ij} + e_{0ij}\text{bcons}^{*} \end{aligned}\right\}$$

$$\text{logit}(\pi_{ij}) = \beta_{1j}\text{cons} + 0.093(0.018)\text{defence}_{ij} + 0.069(0.014)\text{unemp}_{ij} + 0.046(0.019)\text{taxes}_{ij} + 0.143(0.018)\text{privat}_{ij}$$

$$\beta_{1j} = -0.367(0.095) + u_{1j}$$

$$\begin{bmatrix} u_{1j} \end{bmatrix} \sim \text{N}(0, \Omega_u) : \Omega_u = \begin{bmatrix} 0.167(0.119) \end{bmatrix}$$

$$\text{bcons}^{*} = \text{bcons}[\pi_{ij}(1 - \pi_{ij})/\text{denom}_{ij}]^{0.5}$$

$$\begin{bmatrix} e_{0ij} \end{bmatrix} \sim (0, \Omega_e) : \Omega_e = \begin{bmatrix} 1.000(0.000) \end{bmatrix}$$

The fixed-effects parameter estimates are little changed by changing the estimation method, although there is a non-trivial increase in the level 2 variance. All the fixed coefficients are positive, which is consistent with those voters having less liberal attitudes having a higher probability of voting Conservative.

Interpretation of fixed effects

As this model is non-linear the interpretation of the values taken by the fixed effects is more difficult than for a Normal response model. A positive value for a fixed effect will still result in a positive correlation between the value of the predictor and the resulting proportional response, although this relationship will be non-linear. One possible way of interpreting the fixed effects is to consider the effect of increasing a particular predictor on the proportional response for a person with average predictor values.

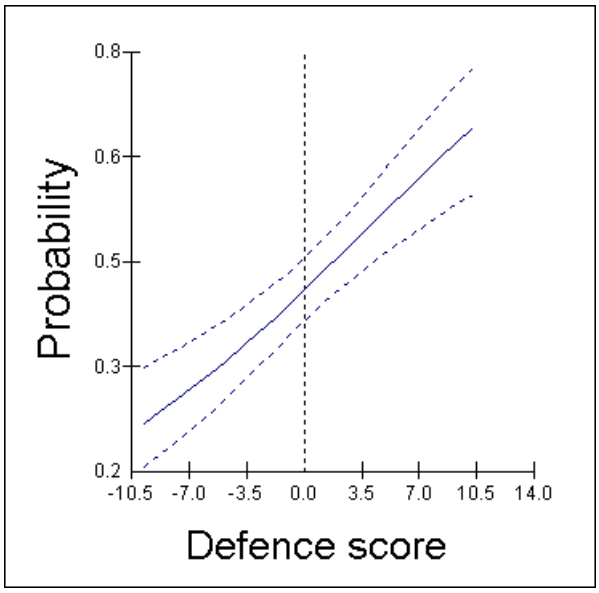
For example if we consider a voter with average views in the voting intentions dataset they have an estimated 40.9% probability of voting Conservative. This is calculated by working out the antilogit function of $X\beta$ when all the predictors have value zero as follows:

$p = [1+\exp(-X\beta)]^{-1} = [1+\exp(0.367)]^{-1} = 0.409$.

You can use the **ALOGit** function in the **Calculate menu** to do this calculation.

This same formula can be used to calculate the effect of a 5 points above average view on possessing nuclear weapons, with the other explanatory variables held at zero. Here

$p=[1+\exp(-X\beta)]^{-1}=[1+\exp(0.367-5*0.093)]^{-1}=0.524$. As the model is non-linear if we consider a 10 points above average view on possessing Nuclear weapons, this will not result in the same increase on the probability scale. In fact $p = [1+\exp(-X\beta)]^{-1} = [1+\exp(0.367 - 10*0.093)]^{-1}= 0.637$ (63.7%) which results in an 11.3% increase as opposed to an 11.5% increase. Plotting the expected probability of voting Conservative against the possible (centred) defence scores (see graph below) also shows that the graph is not a straight line due to the non-linear link function. The graph includes confidence bands and these are not symmetric which is also due to the non-linearity of the link function.



Similar 5 point increases in opinions on unemployment, taxes and privatisation for the average person will result in estimated probabilities of 49.5%, 46.6% and 58.6% of voting Conservative respectively.

From other modelling not presented here there is a slight indication that the coefficient for privatisation (privat) may vary across constituencies. Fitting this parameter as random at level 2 gives the following estimates at level 2:

$$\Omega_u = \begin{bmatrix} 0.232(0.141) & \\ -0.030(0.022) & 0.011(0.006) \end{bmatrix}$$

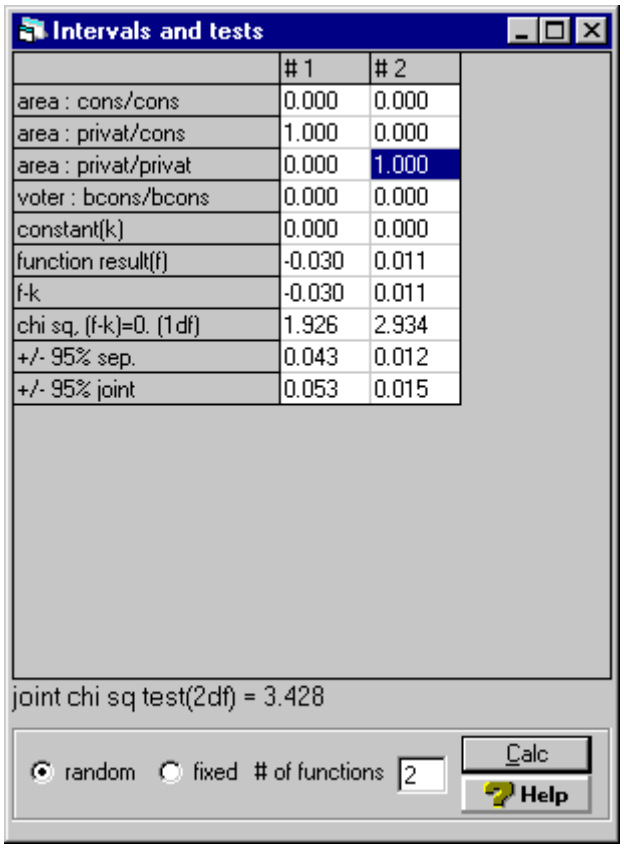
We can carry out an approximate (Wald) hypothesis test for the extra variance and the covariance using the **Intervals and tests** window by clicking on this option under the **Model** menu. The test should be set up as follows, changing the number of functions to 2. Each function is a vector of coefficients which multiplies the fixed or random parameter estimates shown.

If C is this vector then the constraint for the estimated parameter set $\hat{\theta}$ is of the form

$$C^T\hat{\theta} = k$$

where k is a constant (typically zero).

We require two functions here that pick out the variance and covariance term for **privat.** We therefore place a '1' in just these positions as shown below. The



	# 1	# 2
area : cons/cons	0.000	0.000
area : privat/cons	1.000	0.000
area : privat/privat	0.000	1.000
voter : bcons/bcons	0.000	0.000
constant(k)	0.000	0.000
function result(f)	-0.030	0.011
f-k	-0.030	0.011
chi sq. (f-k)=0. (1df)	1.926	2.934
+/- 95% sep.	0.043	0.012
+/- 95% joint	0.053	0.015

This yields a chi-squared value for the joint hypothesis that both parameters are zero of 3.43 on 2 degrees of freedom, providing little evidence for an effect. The interpretation of the separate (sep.) and joint 95% confidence intervals is that for the former a simple Normal distribution interval is computed (based upon 1.96 standard deviations) and for the latter an interval is computed based upon the joint (chi squared) distribution of the constraint functions such that all the intervals include the population values with probability 0.95.

Note that the values you obtain for the parameters and function results may differ very slightly from the above depending on the order in which variables have been entered into the model.

This ends our discussion of binary response models. We now consider fitting models where the response is a proportion, again using the voting intentions dataset.

Models for proportions as responses.

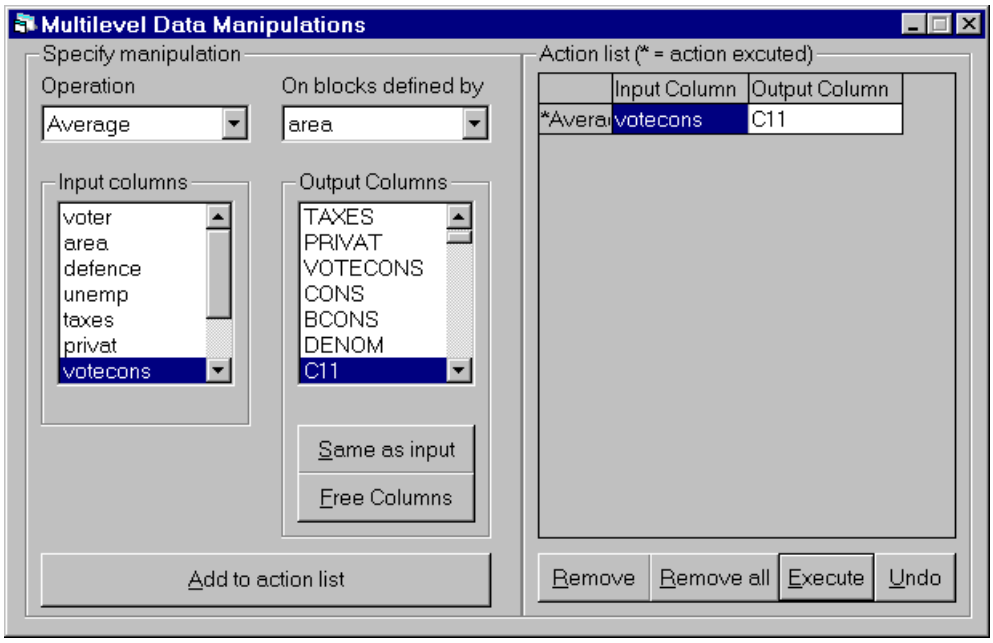
In the voting dataset that we have considered for fitting binary response models, predictor variables have been collected for every single 0-1 response. If this is the case then it will be desirable to fit the response as a 0-1 value using a binary response model.

We fitted two models to the voting intentions dataset, the first of which only included an intercept term. This model does not benefit from considering the individual 0-1 values as the response as no predictors at this level are included and so instead we could have fitted this model as a general Binomial response model.

Fitting the response as a proportion has the advantage that the dataset is reduced in size and so the time to fit such models should decrease. This is particularly useful for the more computationally intensive MCMC and bootstrap methods considered in later chapters. In this section we illustrate how to fit the first model for the voting intentions dataset using a proportional response. This will also illustrate some of the multilevel data manipulation options in *MLwiN* as we have to transform the dataset into the correct format.

Converting the dataset to a proportional response

Release 1.1 of *MLwiN* includes new screens designed to replace the command interface and make data manipulation easier. Firstly we will need to select **Multilevel data manipulations** from the **Data Manipulation** menu. This will bring up a window designed to perform many data manipulation operations that are based on the multilevel structure of the data. We want to create a column that contains the proportion of people voting Conservative in each constituency to use as our new response. To do this we need to select **Average** from the operation box and select **area** as the variable that defines the blocks. Then select **votecons** as the input column and click on **Free columns** to use the first free column (in this case c11) as the output of the operation. Then clicking on **Add to action list** will confirm the operation in the action list on the right of the window. Clicking on **Execute** will perform the action and the screen should then appear as follows:

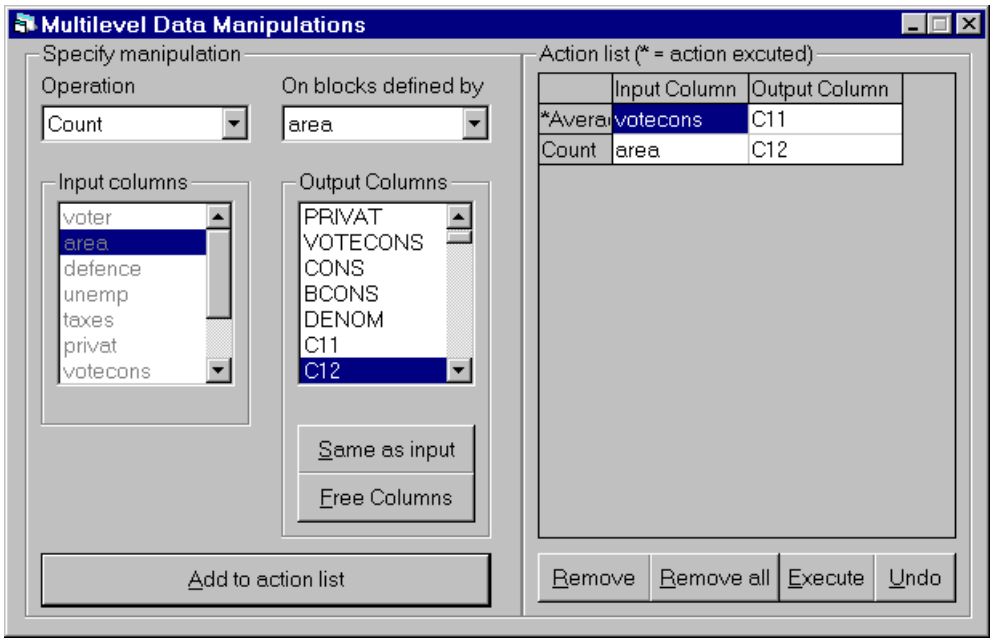


Once the action has been executed it is marked with an asterisk in the Action list.

The operation we have performed by using this window is equivalent to typing the following command in the command interface window:

MLAV ‘area’ ‘votecons’ c11

We have created a new column, c11 that contains for each observation the average proportion of people voting Conservative in the area containing the observation. We also need to create a second column containing the number of observations in each area to replace the **denom** column of 1’s. To do this we need to select **Count** from the operation box and this will then automatically grey out the list of input columns. Then select **area** as the column that defines the blocks and click on **Free columns** to use the next free column (in this case c12) as the output of the command. Then clicking on **Add to action list** will confirm the operation in the action list on the right of the window. The screen should then appear as follows:

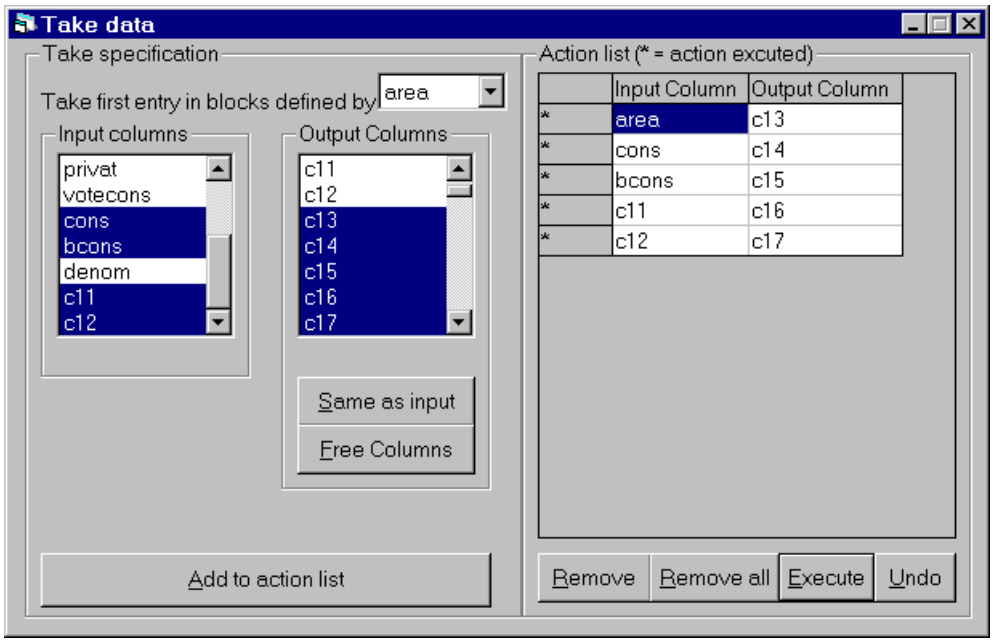


Clicking on **Execute** will then execute the second operation and an asterisk (*) will appear in the action list box. The operation we have performed is equivalent to typing the following command in the command interface window:

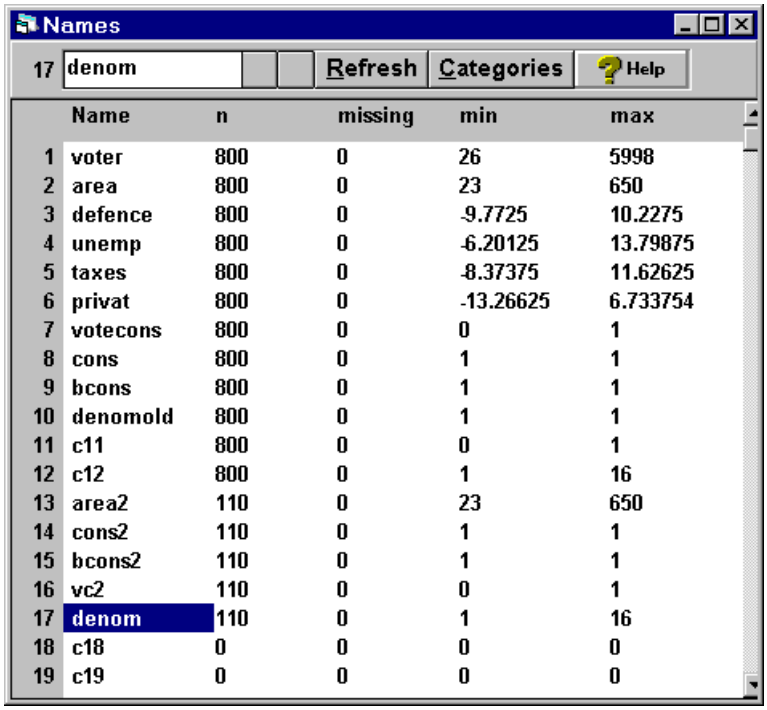
MLCO ‘area’ c12

We have now created our new response and denominator column but they are both of length 800. We now need to reduce these columns to length 110 (number of areas) along with the columns **area, cons** and **bcons**.

To avoid losing our original data we will put these new reduced length variables into new columns. To do this we need to use the **Unreplicate** option from the **Data manipulation** menu. This will bring up a window that is similar in appearance to the multilevel data manipulation window. On this window we will select the variable **area** to define the blocks. Then from the **Input column** list we need to select all the variables that we need for our model, namely **area, cons, bcons**, and the two variables we have just generated that are stored in columns c11 and c12. Click on each of these variables in turn while holding down the control (Ctrl) key to make the multiple selection. We will use the **Free Columns** option to define the output columns and press **Add to action list** to define the action. We then press **Execute** to activate the command. When this is done the window should be as follows:



The new variables will be stored in columns c13-c17 and are of length 110, 1 record for each area. The new variables should now be named by using the **Names** window as shown below:



	Name	n	missing	min	max
1	voter	800	0	26	5998
2	area	800	0	23	650
3	defence	800	0	-9.7725	10.2275
4	unemp	800	0	-6.20125	13.79875
5	taxes	800	0	-8.37375	11.62625
6	privat	800	0	-13.26625	6.733754
7	votecons	800	0	0	1
8	cons	800	0	1	1
9	bcons	800	0	1	1
10	denomold	800	0	1	1
11	c11	800	0	0	1
12	c12	800	0	1	16
13	area2	110	0	23	650
14	cons2	110	0	1	1
15	bcons2	110	0	1	1
16	vc2	110	0	0	1
17	denom	110	0	1	16
18	c18	0	0	0	0
19	c19	0	0	0	0

Note that the old **denom** column should be renamed **denomold** and replaced by a new **denom** column. This is because the name **denom** is a special column recognised by the software macros (as mentioned earlier). Columns C13-C16 are named as shown.

Fitting the model

We now have to specify the names of the new columns in the **Equations** window. First let us remove the old model by pressing the **clear** button on the **equations window** toolbar. Our new response variable is **vc2** and we will use **area2** as both the level 2 IDs and the level 1 IDs. This implies a model with 110 level 2 IDs each with one level 1observation (proportion). This might appear, at first glance, to produce a confounded model, however we should remember that each level 1 unit has an associated denominator which is the number of voters in the area. It is this associated voter-level information that prevents the model from being confounded. To finish the model we must set the predictors **bcons2** as random at level 1 and **cons2** as a fixed effect and random at level 2. Clicking on **START** will produce converged estimates after 4 iterations. The results using first order MQL RIGLS estimation are as shown below:

$$\left.\begin{aligned} \text{vc2}_{ij} &\sim \text{Binomial}(\text{denom}_{ij}, \pi_{ij}) \\ \text{vc2}_{ij} &= \pi_{ij} + e_{0ij}\text{bcons2}^{*} \end{aligned}\right\}$$

$$\text{logit}(\pi_{ij}) = \beta_{1j}\text{cons2}$$

$$\beta_{1j} = -0.248(0.081) + u_{1j}$$

$$\begin{bmatrix} u_{1j} \end{bmatrix} \sim \text{N}(0,\ \Omega_u)\ :\ \Omega_u = \begin{bmatrix} 0.134(0.091) \end{bmatrix}$$

$$\text{bcons2}^{*} = \text{bcons2}[\pi_{ij}(1 - \pi_{ij})/\text{denom}_{ij}]^{0.5}$$

$$\begin{bmatrix} e_{0ij} \end{bmatrix} \sim (0,\ \Omega_e)\ :\ \Omega_e = \begin{bmatrix} 1.000(0.000) \end{bmatrix}$$

As we can see, these estimates are identical to those achieved when the full 800 binary responses were used. This is to be expected as no predictors that vary between individual voters are included in the model.

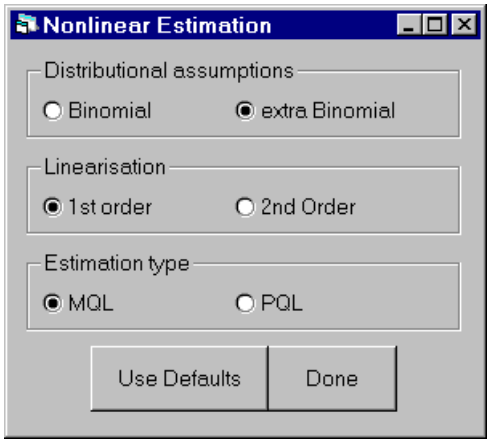
Extra Binomial variation

The binomial distribution is the standard distribution used to fit proportion data. The binomial distribution has an additional constraint when compared with the Normal distribution used for continuous data in that the mean and variance of the distribution are related. In fact, if data are truly Binomially distributed then having found the mean of the distribution, the variance is uniquely defined.

Obviously not all data that consist of proportions are from a Binomial distribution. Sometimes the data exhibit more variation than a Binomial distribution and this is

known as 'over-dispersion', and sometimes the data exhibit less variability and this is known as 'under-dispersion'.

In *MLwiN* it is easy to test whether or not the Binomial assumption holds or whether the model exhibits 'extra Binomial' variation. Considering the model we have just fitted to the voting dataset we click on the **Nonlinear** button to bring up the **Nonlinear Estimation** window. On this window we will leave the estimation method as 1st order MQL but select **extra Binomial** for the distributional assumptions as follows:



Clicking on **Start** will give the following estimates:

$$\left.\begin{aligned} &\text{vc2}_{ij} \sim \text{Binomial}(\text{denom}_{ij}, \pi_{ij}) \\ &\text{vc2}_{ij} = \pi_{ij} + e_{0ij}\text{bcons2}^{*} \end{aligned}\right\}$$

$$\text{logit}(\pi_{ij}) = \beta_{1j}\text{cons2}$$

$$\beta_{1j} = -0.248(0.081) + u_{1j}$$

$$\begin{bmatrix} u_{1j} \end{bmatrix} \sim \text{N}(0,\ \Omega_u) \ :\ \Omega_u = \begin{bmatrix} 0.132(0.219) \end{bmatrix}$$

$$\text{bcons2}^{*} = \text{bcons2}[\pi_{ij}(1 - \pi_{ij})/\text{denom}_{ij}]^{0.5}$$

$$\begin{bmatrix} e_{0ij} \end{bmatrix} \sim (0,\ \Omega_e) \ :\ \Omega_e = \begin{bmatrix} 1.004(0.400) \end{bmatrix}$$

Here we see that the level 1 variance multiplier has increased only very slightly from 1.000 to 1.004 and the other parameters have hardly changed. This means that there is no evidence that our response is not Binomially distributed.

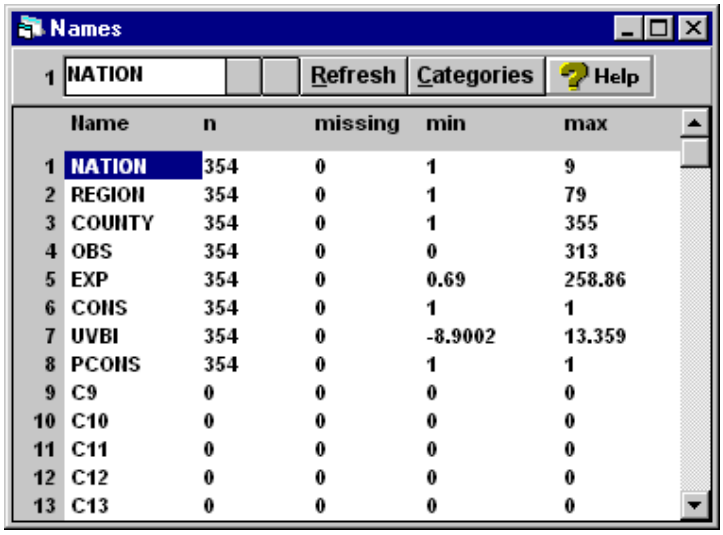
What you should have learnt from this chapter

- How to specify a binary response model in *MLwiN* via the equations window.
- The fact that standard likelihood methods cannot be used for binary response models and so quasi-likelihood methods are used instead.
- The interpretation of fixed effects is more complicated in binary response models.
- How to use the Intervals and Tests window to check if random effects are significant.
- How to fit a general Binomial model where the response is a proportion.
- How to use the multilevel data manipulations options in *MLwiN*.
- How to check the Binomial assumption by fitting extra-Binomial variation.

Chapter 9. Modelling Count Data

Malignant melanoma mortality in the European Community

The example we will consider for fitting a model where the response is a count comes from the field of environmental health. The problem involves assessing the effect of UV radiation exposure on the mortality rate due to malignant melanoma in the European Community and is investigated more thoroughly in Langford, Bentham and McDonald (1998). Open the dataset mmmec.ws using the **Open Worksheet** option from the **Files** menu. Opening the **Names** window from the **Data Manipulation** menu gives the following:



	Name	n	missing	min	max
1	NATION	354	0	1	9
2	REGION	354	0	1	79
3	COUNTY	354	0	1	355
4	OBS	354	0	0	313
5	EXP	354	0	0.69	258.86
6	CONS	354	0	1	1
7	UVBI	354	0	-8.9002	13.359
8	PCONS	354	0	1	1
9	C9	0	0	0	0
10	C10	0	0	0	0
11	C11	0	0	0	0
12	C12	0	0	0	0
13	C13	0	0	0	0

The response variable (obs) in this dataset is the number of male deaths due to malignant melanoma in various regions of the European community during the period 1971 to 1980. The data were collected in 9 countries, and these countries were each subdivided into regions which were in turn subdivided into counties to give three levels of data.

The variable **exp** is the expected number of deaths in each county which is proportional to the population size in each county and is used to create an offset variable as will be explained later. The variable **uvbi** is a measure of the UVB dose reaching the earth's surface in each county and this variable has been centred. The variables **cons** and **pcons** are both columns of 1's; **pcons** is used for the level 1 variation in a similar way to **bcons** in the binomial response examples.

Fitting Poisson Models

Count data are constrained to be non-negative but if we fit a Normal model to the data we could produce predicted counts that were negative and so we would prefer to model the logarithms of the counts. We will therefore fit a Poisson model to the count data using a log link function. We are actually more interested in the rates of malignant

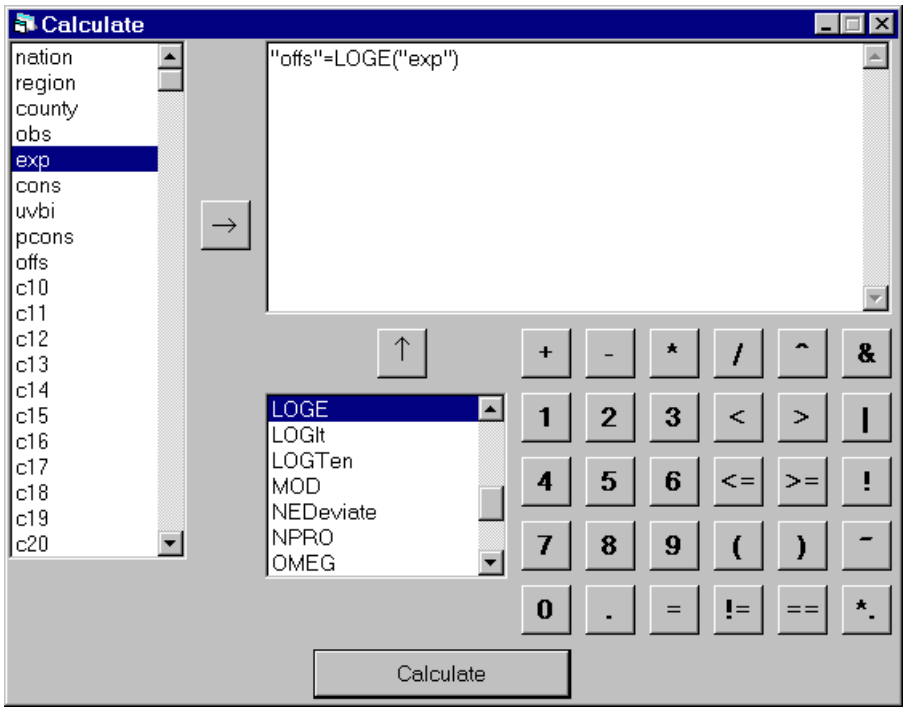
melanoma mortality rather than the actual counts as each geographic unit will have a different population size. Therefore if we were to use the raw counts, the units with larger population size will have larger counts thus masking the true relationship. To work with the rates rather than the counts we use an additional parameter known as an offset.

This offset is set to be equal to the log (base *e*) of the expected count (which is based on unit size) and so adding this term leads to the general Poisson multilevel model:

$y \sim \text{Poisson}(\pi)$

$\log(\pi) = \log(exp) + X\beta + Z\theta$, or alternatively $\log(\pi/exp) = X\beta + Z\theta$.

To include an offset in *MLwiN*, we must include a column called 'offs' that the software identifies as a special column including the offset terms. Note that the 'offs' column should not be added to the model via the **Equations** window as it is included automatically in the model as long as it exists in the worksheet. We will use the **Names** window to name column c9 as "offs" and then to calculate the offsets we use the **Calculate** window as follows:



We will firstly ignore the structure of the data and fit a single level Poisson model. To fit a single level model in *MLwiN* we must actually set up a two level model using 'cons' to define a single level 2 block but not include any terms random at level 2. To set up the response variable and level identifiers click on *y* in the equations window and

select 'obs' as the response, 2 as the number of levels, 'cons' as level 2 identifier and 'county' as the level 1 identifier.

To run a Poisson model we need to click on the *N* that appears by default and select Poisson. We must also click on 'logit' and select instead the log link function. The log link is the standard link function for Poisson models as it transforms the response variable, which must be positive, into a variable that is defined on the whole real line. As with the Binomial models a second constant column, which we have called 'pcons' here needs to be created. This column is used to calculate the level 1 variation and should be included as random at level 1 only. We now include 'cons' and 'uvbi' as fixed effects and click on the **Names** button on the **Equations** window. Then running the model using the 1st order MQL estimation method we should get the following results upon convergence:

$$\left.\begin{aligned} \text{obs}_{ij} &\sim \text{Poisson}(\pi_{ij}) \\ \text{obs}_{ij} &= \pi_{ij} + e_{0ij}\text{pcons}^{*} \end{aligned}\right\}$$

$$\log(\pi_{ij}) = \underset{(0.011)}{-0.070}\text{cons} + \underset{(0.003)}{-0.057}\text{uvbi}_{ij}$$

$$\text{pcons}^{*} = \text{pcons}\,\pi_{ij}^{0.5}$$

$$\left[e_{0ij}\right] \sim (0,\ \Omega_e) : \Omega_e = \underset{(0.000)}{\left[1.000\right]}$$

Note that line 4 of the above equations reflects the fact that the variance of a Poisson variable with mean π is also π. We can see here that in this model there is a negative relationship between incidence of melanoma and UV exposure. This seems surprising but may be explained by including more structure into the data. Note also, that as with binomial models the default variance of the e_{0ij}'s is 1.0. This denotes Poisson variation but can be unconstrained to allow extra Poisson variation.

We will now consider the dataset as a 3 level model and so we need to change the level ids to 'nation' for level 3, 'region' for level 2 and 'county' for level 1. The first model we wish to consider is a simple variance components model to consider the geographical effects on melanoma mortality without adjusting for UV exposure. To do this we remove 'uvbi' from the model. 'Cons' is set as random at both the 'nation' and 'region' levels.

This model was run using RIGLS and 1st order MQL and the results are as follows:

$$\left.\begin{aligned} &\text{obs}_{ijk} \sim \text{Poisson}(\pi_{ijk}) \\ &\text{obs}_{ijk} = \pi_{ijk} + e_{0ijk}\text{pcons}^{*} \end{aligned}\right\}$$

$$\log(\pi_{ijk}) = \beta_{1jk}\text{cons}$$

$$\beta_{1jk} = 0.110(0.160) + v_{1k} + u_{1jk}$$

$$\begin{bmatrix} v_{1k} \end{bmatrix} \sim \text{N}(0,\ \Omega_v)\ :\ \Omega_v = \begin{bmatrix} 0.214(0.109) \end{bmatrix}$$

$$\begin{bmatrix} u_{1jk} \end{bmatrix} \sim \text{N}(0,\ \Omega_u)\ :\ \Omega_u = \begin{bmatrix} 0.045(0.010) \end{bmatrix}$$

$$\text{pcons}^{*} = \text{pcons}\,\pi_{ijk}^{0.5}$$

$$\begin{bmatrix} e_{0ijk} \end{bmatrix} \sim (0,\ \Omega_e)\ :\ \Omega_e = \begin{bmatrix} 1.000(0.000) \end{bmatrix}$$

This model shows that there is nearly five times as much variability between nations as there is between regions within nations. The 1st order MQL method is not as accurate as the 2nd order PQL method so we will now fit the same model with this method:

$$\left.\begin{aligned} &\text{obs}_{ijk} \sim \text{Poisson}(\pi_{ijk}) \\ &\text{obs}_{ijk} = \pi_{ijk} + e_{0ijk}\text{pcons}^{*} \end{aligned}\right\}$$

$$\log(\pi_{ijk}) = \beta_{1jk}\text{cons}$$

$$\beta_{1jk} = -0.024(0.152) + v_{1k} + u_{1jk}$$

$$\begin{bmatrix} v_{1k} \end{bmatrix} \sim \text{N}(0,\ \Omega_v)\ :\ \Omega_v = \begin{bmatrix} 0.185(0.097) \end{bmatrix}$$

$$\begin{bmatrix} u_{1jk} \end{bmatrix} \sim \text{N}(0,\ \Omega_u)\ :\ \Omega_u = \begin{bmatrix} 0.058(0.012) \end{bmatrix}$$

$$\text{pcons}^{*} = \text{pcons}\,\pi_{ijk}^{0.5}$$

$$\begin{bmatrix} e_{0ijk} \end{bmatrix} \sim (0,\ \Omega_e)\ :\ \Omega_e = \begin{bmatrix} 1.000(0.000) \end{bmatrix}$$

The results for 2nd order PQL are similar to the MQL results but with a different split of the variance between the two levels. One important difference with Poisson models that has also been shown by some simulations is that the MQL method tends to overestimate some of the variance parameters. This is in contrast with other types of discrete response model where this method tends to underestimate the variance. For the

remainder of the models in this chapter we will only consider the 2nd order PQL method.

The next step in our model fitting with this dataset is to add the predictor 'uvbi' into the multilevel model. The results of using 2nd order PQL on this model are as follows:

$$\left.\begin{aligned} \text{obs}_{ijk} &\sim \text{Poisson}(\pi_{ijk}) \\ \text{obs}_{ijk} &= \pi_{ijk} + e_{0ijk}\text{pcons}^{*} \end{aligned}\right\}$$

$$\log(\pi_{ijk}) = \beta_{1jk}\text{cons} + \text{-0.028(0.011)}\text{uvbi}_{ijk}$$

$$\beta_{1jk} = \text{-0.082(0.142)} + v_{1k} + u_{1jk}$$

$$\begin{bmatrix} v_{1k} \end{bmatrix} \sim \text{N}(0,\ \Omega_v) \ :\ \Omega_v = \begin{bmatrix} \text{0.157(0.083)} \end{bmatrix}$$

$$\begin{bmatrix} u_{1jk} \end{bmatrix} \sim \text{N}(0,\ \Omega_u) \ :\ \Omega_u = \begin{bmatrix} \text{0.049(0.011)} \end{bmatrix}$$

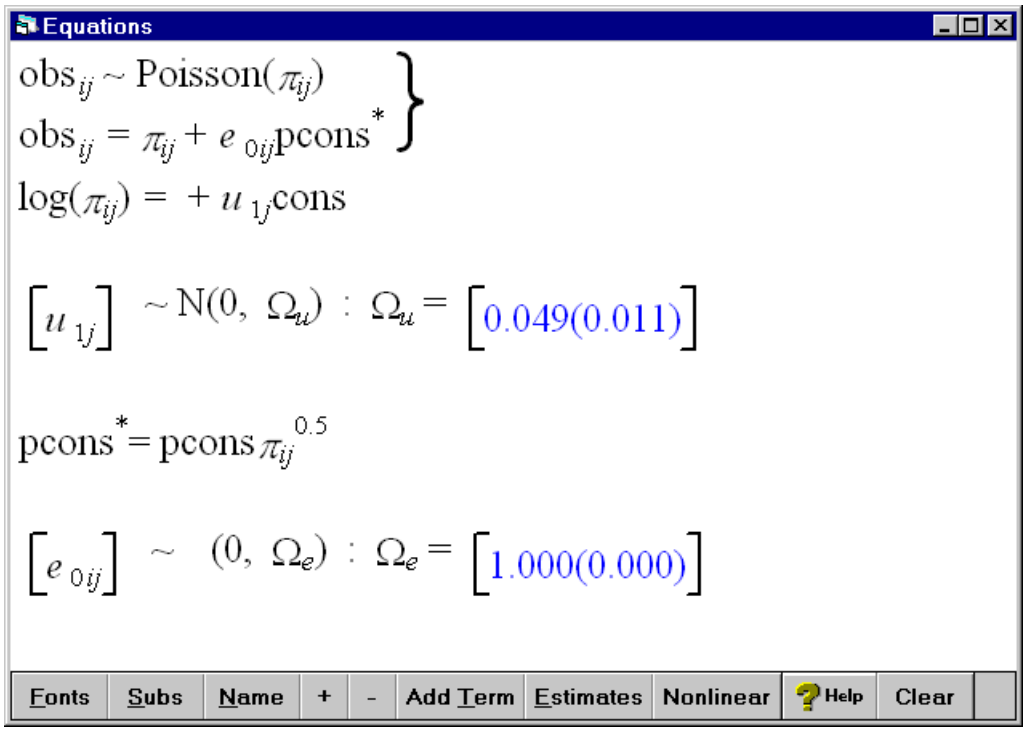
$$\text{pcons}^{*} = \text{pcons}\,\pi_{ijk}^{0.5}$$

$$\begin{bmatrix} e_{0ijk} \end{bmatrix} \sim (0,\ \Omega_e) \ :\ \Omega_e = \begin{bmatrix} \text{1.000(0.000)} \end{bmatrix}$$

This model again shows that the variability between nations is three times the variability between the regions within nations. The amount of UV radiation still has a significant negative effect on melanoma mortality. In this three level model we have only 9 countries at level 3 and so the accuracy of the level 3 variance estimate is low. An alternative model that could be used would consist of fitting separate fixed terms for each country by including country indicators in the fixed part of the model.

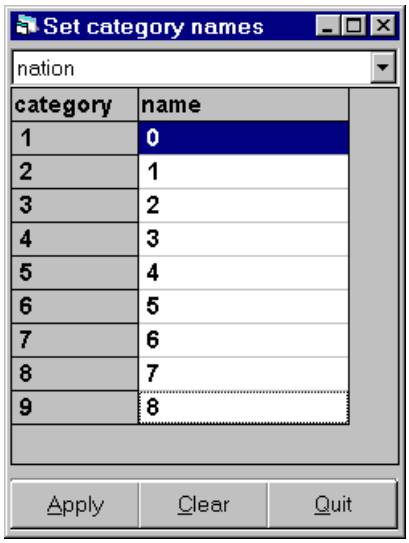
Using Dummy Variables

We can use the **Main effects and interactions** window to set up this model. However, we first must remove '*cons'* and '*uvbi'* as fixed effects from the model, and remove the third level (*nation*) from the model. This produces the following **Equations** window:

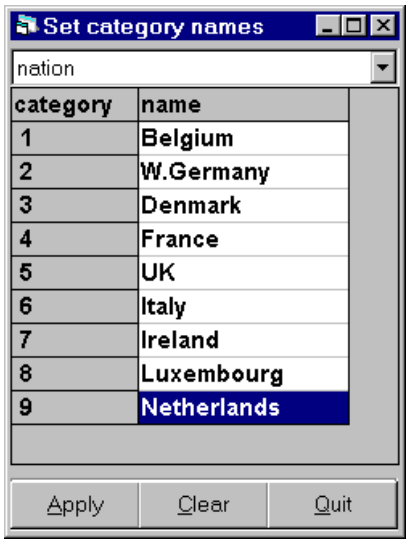


Now we would like to set up nation as a categorical variable rather than a level indicator.

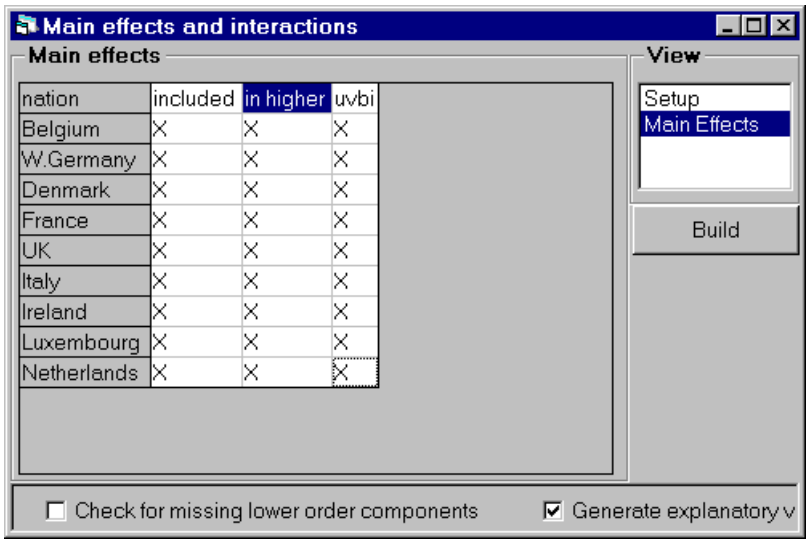
To do this we will use the **Names** window. Click on the Categories button and this will bring up the **Set category names** window as shown below:



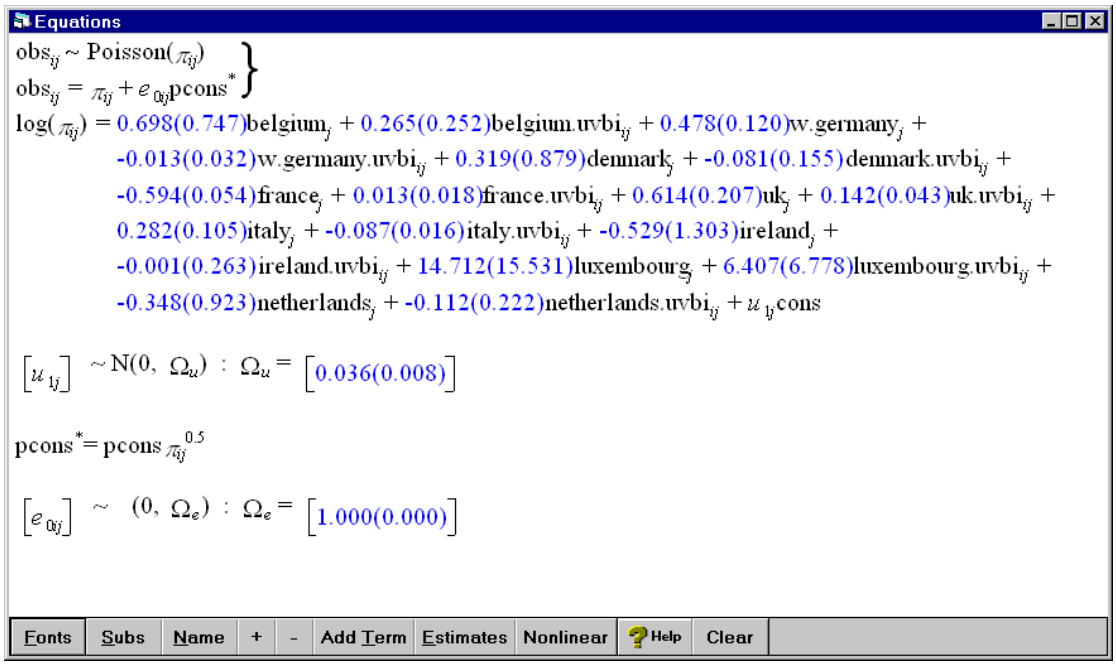
We will now type in the country names into this window as shown below:



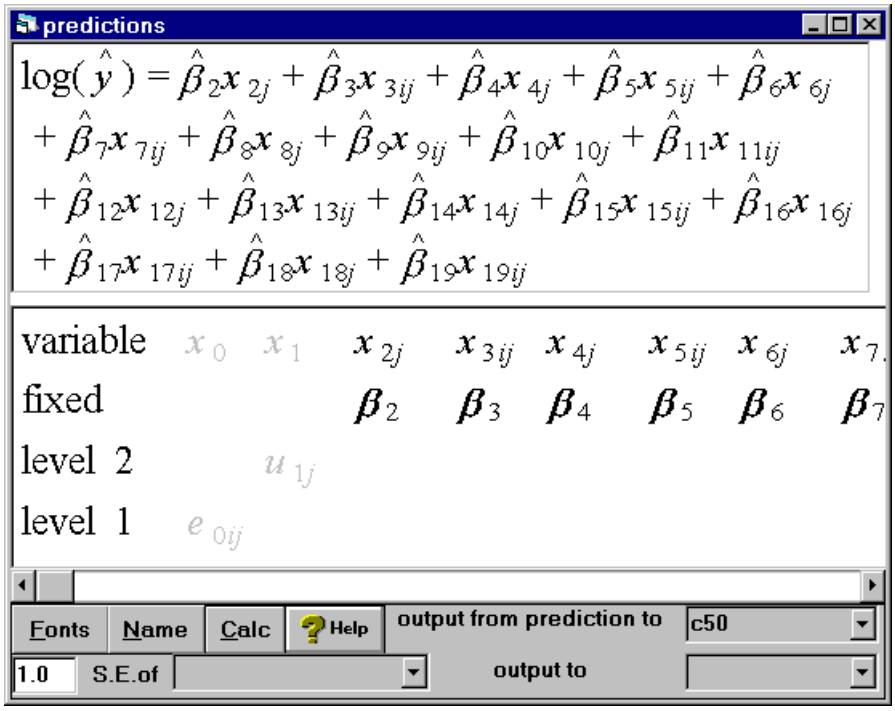
Clicking on **Apply** will now store the category names for the 'nation' variable. Next we can set up the nine separate nation level Poisson regressions using the **Main effects and interactions** window available from the M**odel** menu. In the **Main effects and interactions** window, from the **Setup** options select **nation** in the list of categorical variables and select **uvbi** in the list of continuous variables. Select **Main effects** from the view panel and set up the main effects window with the following pattern:



Now click on the **Build** button and the main effects and interactions will be added to the model. Running this new model using RIGLS 2nd order PQL produces the following results:



We can now form predicted lines for each of the fifteen countries by selecting the **Predictions** window from the **Model** menu, clicking on the word **fixed** in the lower panel and selecting **include all fixed coefficients** from the resulting menu. Select c50 as the output column for the predictions. The **Predictions** window should now be as follows (we have not shown all of it):

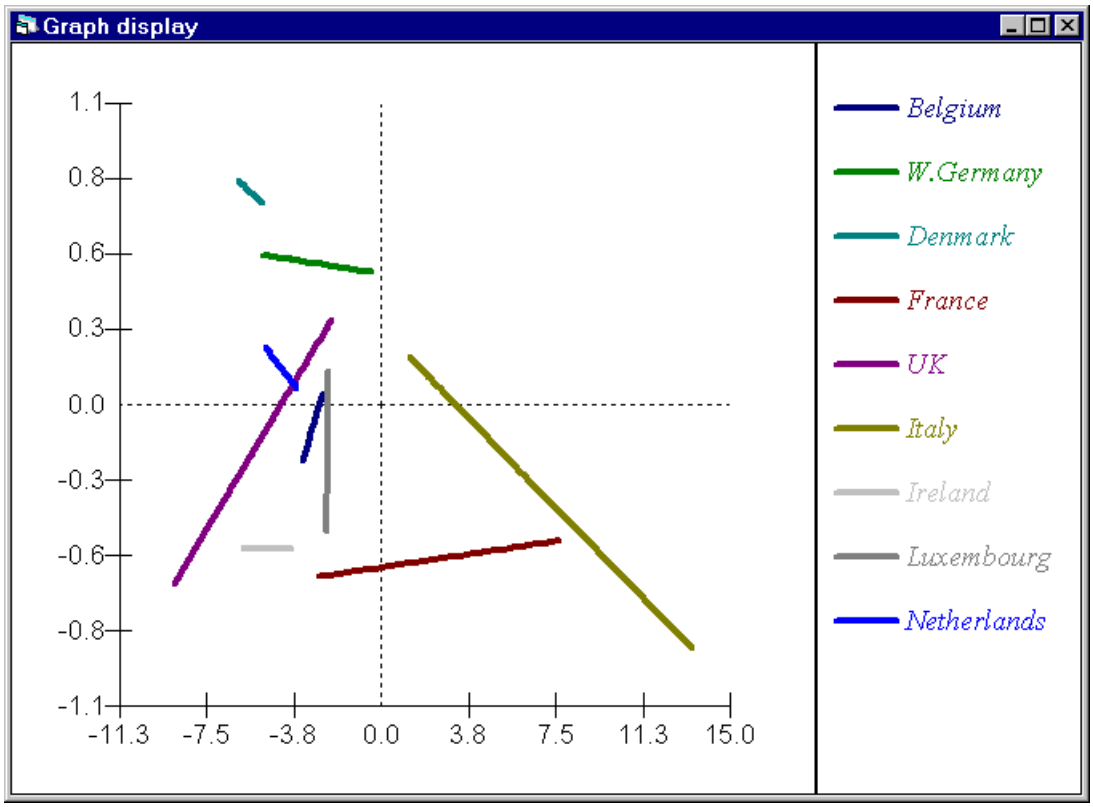


Click on **Calc** to store the predicted values in c50.

We would now like to plot out 9 lines, one for each country. To do this we need to open the **Customised graphs** window from the **Graphs** menu. From the window the following settings should be chosen

- Select **y** as c50 and **x** as **uvbi**.
- Select **group** as **nation**.
- Select **plot type** as **line**.
- On the **plot style** tab select **colour** as 16 rotate and **line thickness** as 4.
- on the **other** tab on the **construct key labels** from area, check **group** code to construct a key.
- Press **Apply**.

Selecting all these options should produce the following graph:



On your computer screen you will be able to identify the lines according to the colour coding. Remember that the predicted values are the logarithm to the base *e* of the relative risk. Here effects of the UV radiation appear to vary dramatically from nation to nation. From the estimates in the equations window we saw that all countries except country 5 (UK) and country 6 (Italy) have estimated effects that are not significantly different from zero. This graph shows a clearer picture of the actual effects of uvbi in each country as we can clearly see that there is very little overlap in terms of uvbi values between some countries.

Luxembourg (country 8) is poorly estimated due to containing only 3 counties and we can see that even though the intercept term for Luxembourg is huge, in fact there are no people in Luxembourg who experience the mean UV exposure (that the intercept represents). For the values of UV exposure experienced in Luxembourg the relative risk is close to zero. The UK has a strong positive association between UV radiation and melanoma mortality and this could be explained in many ways. One reason could be the combination of few hot sunny days at home combined with more recreational travel to warmer climates. Italy on the other hand has a negative association between UV radiation and melanoma mortality. This could be potentially explained by a higher prevalence of low risk dark skinned people in the south of Italy which has a higher UV exposure.

This model ends our examination of the melanoma mortality dataset. To finish the chapter we will consider some general issues involving discrete response models.

Some issues and problems for discrete response models

The binomial response models of the previous chapter and the count data response models of the present chapter are both examples of multilevel *Generalised linear models* **(**McCullagh and Nelder, 1989**)**.

In addition to fitting a Poisson error model for count data we can also fit a negative binomial error which allows greater flexibility, essentially allowing a more complex variance structure than that associated with the Poisson distribution. The negative binomial appears as an extra option on the drop down menu list for error distribution. See Goldstein (1995, Chapter 7).

There are several problems in fitting such generalised linear models, and some of these have been touched upon in this chapter. Research is being actively carried out in this area and future releases of *MLwiN* will reflect this. It will be shown how to fit Generalised linear models using MCMC methods and bootstrapping in later chapters. In general it is recommended that more than one approach is tried, and if similar results are obtained from the various estimation methods then the analyst can have some confidence in the estimates.

Some care is also needed with using any of the standard diagnostic procedures based on estimated residuals in binary response models. Where there are few level 1 units per level 2 unit and/or the underlying probabilities are close to 0 or 1, then these estimates are not approximately Normally distributed, even when the model is correct.

What you should have learnt from this chapter

- How to fit Poisson models to count data.
- How to use an offset in *MLwiN* to model rates rather than raw counts.
- How to fit single level models in *MLwiN*.
- How to define categorical variables in *MLwiN*.
- How to use dummy variables to reduce the number of levels in a model when there are few higher level units.

Chapter 10: Repeated measures data

Repeated measures data arise in a number of contexts, such as child or animal growth, panel surveys and the like. The basic structure is that of measurements nested within subjects, i.e. a 2-level hierarchy.

In *MLwiN* version 1.1 onwards we no longer need to use the **Command interface** window to specify parts of the model.

To begin with, suppose we have a sample of students whose reading attainment is measured on a number of occasions. The students define level two, the repeated measures or occasions define level one. In longitudinal repeated measures designs, we usually have a large number of level two units with rather few level one units each, unlike the design for the examination data in the introductory tutorial. We can, of course, extend this structure to include a third level representing the fact that the students will be in schools. It is also worth bearing in mind that our repeated measures could be of schools or teachers rather than (or even as well as) of students. So we might have a four level structure, with a sample of schools, studied over time by measuring repeated cohorts of students, these students themselves repeatedly measured as they pass through the school. This is clearly likely to be a large study, but one with potential for looking at the stability of school effects as well as for studying students' educational growth.

In the following sections we introduce the data set, and formulate and analyse a sequence of models of increasing complexity. We shall, however, only cover some of the possible elaborations: these models can be extended to the case of complex serial correlation structures at level 1, and to the multivariate case. We shall also not deal with the case of discrete, for example binary, responses since this raises some new issues which are have been studied and can be handled using macros described in the advanced modelling manual (Yang et al., 1999).

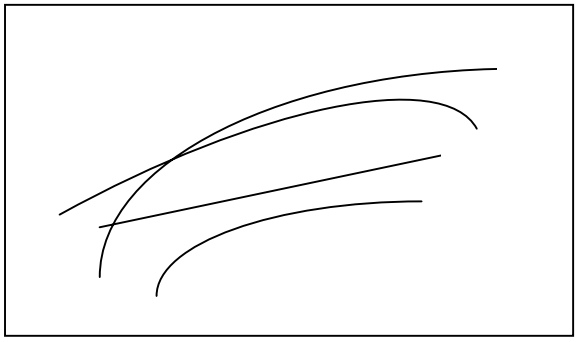
Statistical models for repeated measures data

In multilevel structures we do not require balanced data to obtain efficient estimates. In other words it is not necessary to have the same number of lower level units within each higher level unit, and with repeated measures data we do not require the same number of measurement occasions per individual subject (level 2). Often in longitudinal studies individuals leave the study or miss one or more occasions. Nevertheless, all of the available data can be incorporated into the analysis. This assumes that the probability of being missing is independent of any of the random variables in the model. This condition, known as *completely random dropout* (CRD) may be relaxed to that of *random dropout* (RD) where the missing mechanism depends on the observed measurements. In this latter case, so long as a full information estimation procedure is used, such as that of maximum likelihood in *MLwiN* for

Normal data, then the actual missingness mechanism can be ignored. See Diggle and Kenward (1994) for a discussion of this issue. The ability to handle unbalanced data is in contrast to analyses based upon 'repeated measures analysis of variance' (See Plewis, 1997, Chapter 4).

We can adopt two perspectives on repeated measures data. The first is concerned with what we may term 'growth curves' since such models were originally developed to fit human and animal anthropometric data (See Goldstein, 1979). Some examples are illustrated in figure 10.1.

Figure 10.1. Some examples of growth curves



Time

This shows different kinds of change in a variable with time or age: growth is linear over the age range for case B, non-linear but monotonic for cases D and A, non-linear and non-monotonic for case C.

A second way of looking at repeated measures data, when there is a small number of fixed occasions, is to use a 'conditional' model where later measures are related to one or more earlier measures. This can be a useful approach in some circumstances and it raises no new multilevel modelling issues. In this tutorial we are only concerned with the first type of model, that for repeated measures growth curve data. We now look at an example data set and analysis.

Repeated measures data on reading attainment

The data we are going to use come from a longitudinal study of a cohort of students who entered 33 multi-ethnic inner London infant schools in 1982, and who were followed until the end of their junior schooling in 1989. More details about the study can be found in Tizard et al. (1988). Students' reading attainments were tested on up to six occasions; annually from 1982 to 1986 and in 1989. Reading attainment is the response, and there are three levels of data - school (level 3), student (level 2) and measurement occasion (level 1). In addition, there are three explanatory variables. The

first is the student's age which varies from occasion to occasion and is therefore a level one variable. The other two are gender (coded 0 for males and 1 for females) and ethnic group (coded 0 for white and 1 for black) which vary from student to student and are thus level two variables. The initial sample at school entry was 277, of whom 171 were white indigenous students and 106 black British students of African Caribbean origin. This rose to 371 one year later and fell to 198 by the end of junior school. Some basic questions are:

(i) How does reading attainment change as students get older?

(ii) Does this vary from student to student?

(iii) Does the pattern of change vary from one type of student (for example, girls) to another (boys)?

In this chapter the first two questions will be explored in the following analyses.

Table 10.1 below gives the number of reading tests per student and shows that only a minority of students were measured at every occasion. Altogether, 1758 observations were obtained on 407 students, of whom 259 were white and 148 were black. It is important to note that students with, say, a total of three tests did not necessarily all have tests at the same three occasions. Table 10.2 illustrates some of the response patterns across students. This table also indicates that students' ages can vary at fixed measurement occasions - compare, for example, student one and student four at occasion one. This underlines the other advantage of multilevel modelling of repeated measures which has already been mentioned, namely the ability to handle unequal measurement intervals.

Table 10.1. Summary of Reading Test Data

Number of Tests	Number of Students	% of Total Students	Inverse Cumulative %
1	37	9	100
2	41	10	91
3	42	10	81
4	48	12	71
5	113	28	59
6	126	31	31
TOTAL	407	100	n.a.

Table 10.2. Different Patterns of test sequences and ages of testing

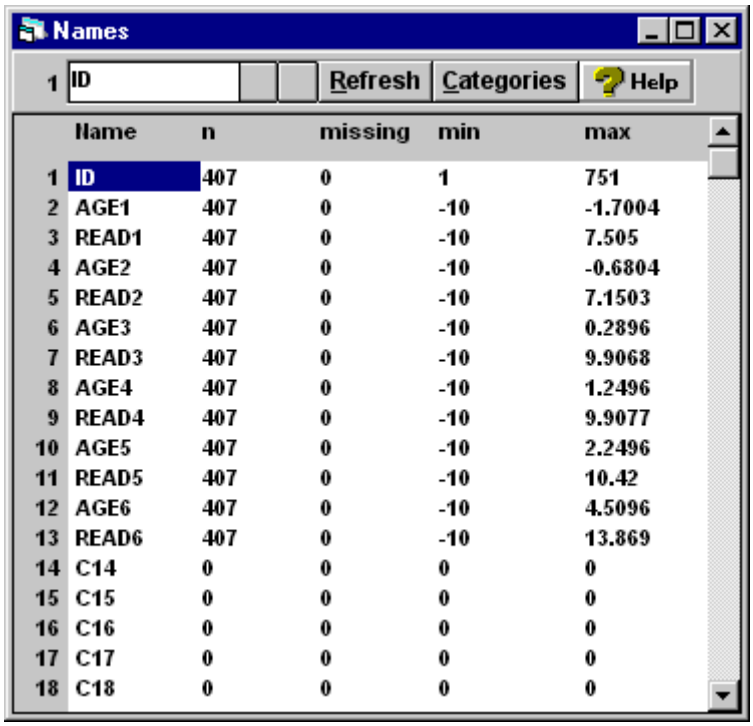
STUDENT	OCCASION					
	1	2	3	4	5	6
ONE	4.6	5.7	6.7	7.7	8.7	11.6
TWO	4.8	5.7	6.8	-	-	-
THREE	4.8	5.8	-	7.8	-	-
FOUR	4.9	-	-	-	-	-

Defining the response variable scale

One problem we have with the present data is just how to define and construct the response over age. Reading attainment cannot usually be measured with the same test at each age, and in our example four rather different, age-appropriate tests were used. The underlying construct is reading but the observed variable changes with age and we wish to construct a common age scale with sensible properties. Moreover, we may find that our results vary as we change the scale of our response. Here we work with a scale for the response defined in the following 'age-equivalent' way. The mean reading score at each occasion is set equal to the mean student age for that occasion, and the variance is set to increase from one occasion to the next in such a way that the coefficient of variation (i.e. the standard deviation divided by the mean) is constant and equal to 0.13. Allowing the variance, as well as the mean, to increase with age is consistent with what we know about many kinds of growth. An alternative measure would be z scores (zero mean and unit variance) at each occasion.

Setting up the data structure

Open the supplied worksheet in *MLwiN*, Reading1.ws, which contains 13 variables for 407 students as shown below in the **Data Manipulation/Names** display:



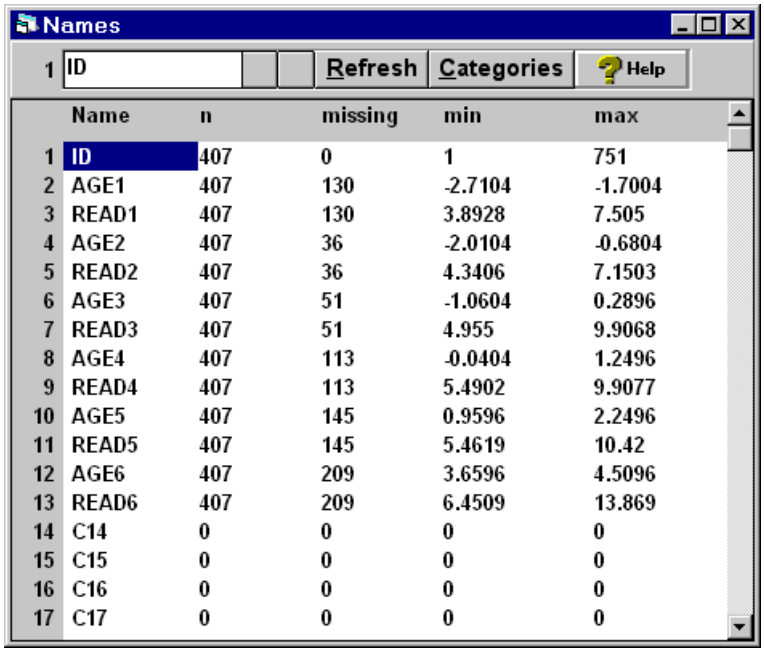
	Name	n	missing	min	max
1	ID	407	0	1	751
2	AGE1	407	0	-10	-1.7004
3	READ1	407	0	-10	7.505
4	AGE2	407	0	-10	-0.6804
5	READ2	407	0	-10	7.1503
6	AGE3	407	0	-10	0.2896
7	READ3	407	0	-10	9.9068
8	AGE4	407	0	-10	1.2496
9	READ4	407	0	-10	9.9077
10	AGE5	407	0	-10	2.2496
11	READ5	407	0	-10	10.42
12	AGE6	407	0	-10	4.5096
13	READ6	407	0	-10	13.869
14	C14	0	0	0	0
15	C15	0	0	0	0
16	C16	0	0	0	0
17	C17	0	0	0	0
18	C18	0	0	0	0

Field or Column number 1 is the student identifier; this is followed by 6 pairs of fields corresponding to the 6 occasions, each pair being the student's reading score and age at the test on that occasion. Note that the ages have been centred around the mean age and

–10 in this data set represents missing data. We can tell *MLwiN* that –10 is the missing value code by

Select the **options** menu Select **Numbers(display precision and missing value code)** Set **missing value** code to –10 Press the **Apply** button, then **Done**

The **names** window is updated and now explicitly shows the number of missing cases in each variable.



	Name	n	missing	min	max
1	ID	407	0	1	751
2	AGE1	407	130	-2.7104	-1.7004
3	READ1	407	130	3.8928	7.505
4	AGE2	407	36	-2.0104	-0.6804
5	READ2	407	36	4.3406	7.1503
6	AGE3	407	51	-1.0604	0.2896
7	READ3	407	51	4.955	9.9068
8	AGE4	407	113	-0.0404	1.2496
9	READ4	407	113	5.4902	9.9077
10	AGE5	407	145	0.9596	2.2496
11	READ5	407	145	5.4619	10.42
12	AGE6	407	209	3.6596	4.5096
13	READ6	407	209	6.4509	13.869
14	C14	0	0	0	0
15	C15	0	0	0	0
16	C16	0	0	0	0
17	C17	0	0	0	0

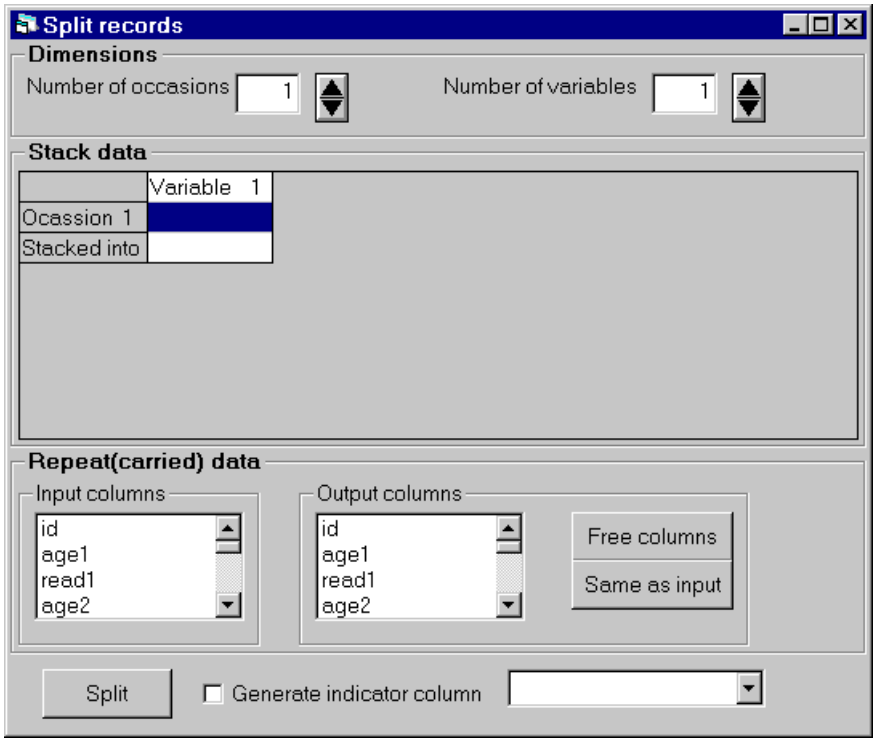
This arrangement of data, in which each row of a rectangular array corresponds to a different individual and contains all the data for that individual, is a natural one, but it does not reflect the hierarchical structure of measurements nested within individuals. The **Split records** window available on the **data manipulation** menu is designed to produce separate records (or rows), one for each occasion. In the present case we shall produce 6 records per student, that is, 2442 records altogether. The ordering of students will be preserved, and they will become the level 2 units.

There are two things to consider:

- Occasion specific data - the data which (in principle) change from occasion to occasion, in this case, the reading scores and the ages.

- Repeated data - the data which remain constant from occasion to occasion, in this case, the student identifiers.

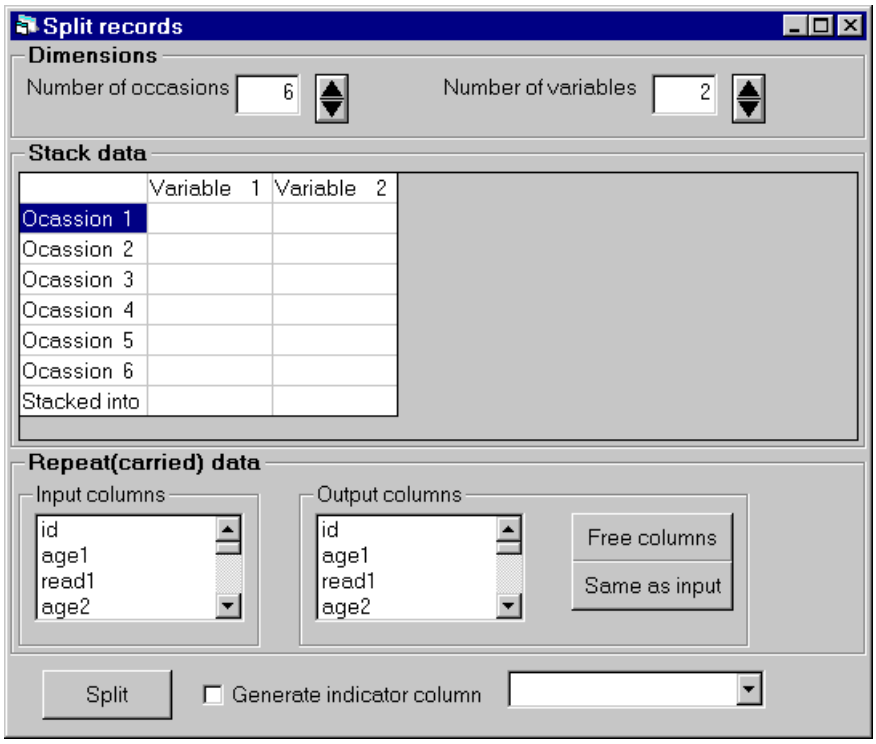
Bring up the **split records** window, under **data manipulation** :



First let us deal with the occasion specific data :

Set the number of occasions to 6 Set the number of variables to 2

Which produces :



We need to stack up the six reading scores into a single column and the six ages into a single column.

In the **Stack data** grid click on **variable 1**

From the drop down list that appears select the six variables **read1-read6** (to make multiple selections hold the control key down while clicking on variable names)

Repeat the above two steps for **variable 2** and the six variables **age1-age6**

Clicking on the column headings allows you to set all 6 occasion variables from a single pick list. The first variable on the list is assigned to occasion 1, the second to occasion 2 and so on. This works fine in our case because the variables appear on the list in the correct order. If this is not the case you can specifically assign variables to occasions by clicking on individual cells in the grid.

Click in turn on the two empty cells in the **stacked into** row of the **Stack data** grid

From the drop down lists that appear select **c14** and **c15** respectively.

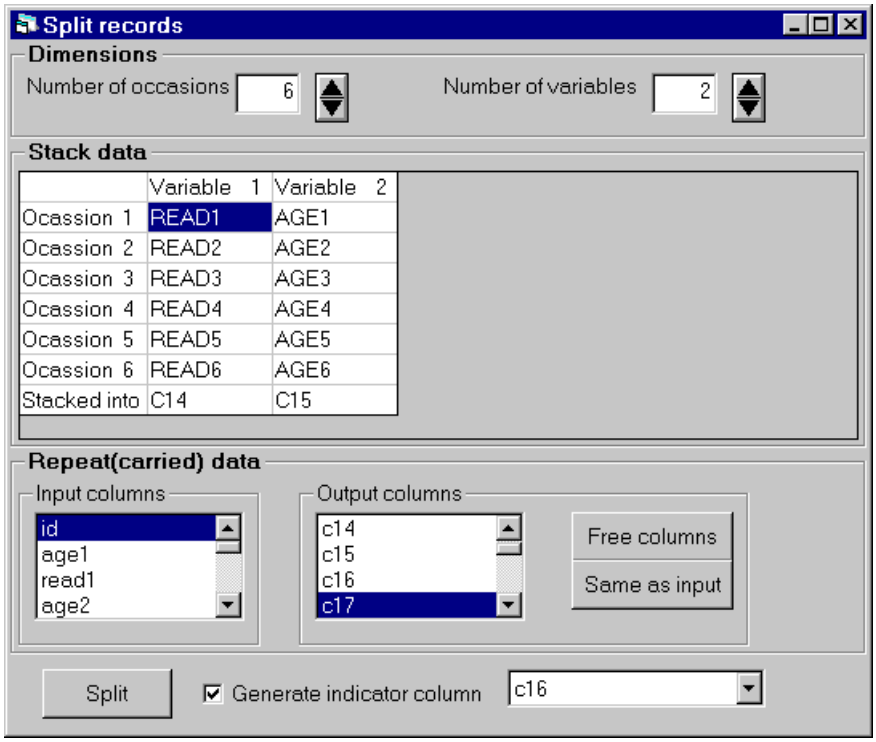
Tick the **generate indicator column** check box

In neighbouring drop down list select **c16**

That deals with occasion specific data. Now for the repeated data :

In the **Repeat(carried data)** frame select **id** as the input column and **c17** as the output column

The completed screen should look like this :

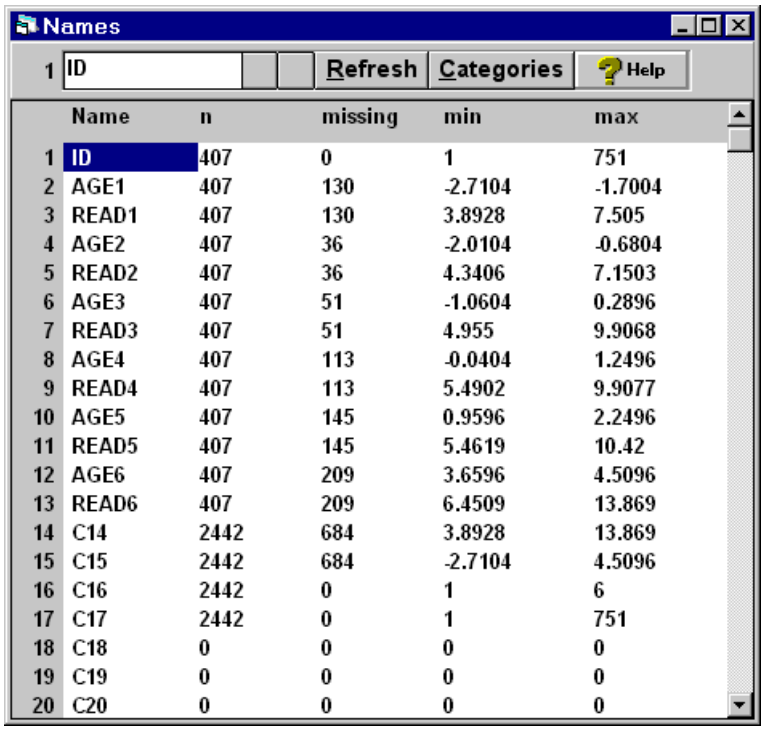


This will take the 6 reading score variables each length 407 and stack them into a single variable in column 14. The six age variables will be stacked into column 15. Each **id** code will be repeated six times and the repeated codes are stored in column 17. The indicator column which is output to column 16 will contain occasion identifiers for the new long data set. The output columns 14-17 will contain 6*407=2442 records.

Press the **Split** button to execute the changes

You will be asked if you want to save the worksheet – select NO

The names window now looks like this



	Name	n	missing	min	max
1	ID	407	0	1	751
2	AGE1	407	130	-2.7104	-1.7004
3	READ1	407	130	3.8928	7.505
4	AGE2	407	36	-2.0104	-0.6804
5	READ2	407	36	4.3406	7.1503
6	AGE3	407	51	-1.0604	0.2896
7	READ3	407	51	4.955	9.9068
8	AGE4	407	113	-0.0404	1.2496
9	READ4	407	113	5.4902	9.9077
10	AGE5	407	145	0.9596	2.2496
11	READ5	407	145	5.4619	10.42
12	AGE6	407	209	3.6596	4.5096
13	READ6	407	209	6.4509	13.869
14	C14	2442	684	3.8928	13.869
15	C15	2442	684	-2.7104	4.5096
16	C16	2442	0	1	6
17	C17	2442	0	1	751
18	C18	0	0	0	0
19	C19	0	0	0	0
20	C20	0	0	0	0

Name columns 14-17 to be **reading, age, occasion** and **student.** Viewing columns 14-17 will now show :

Data

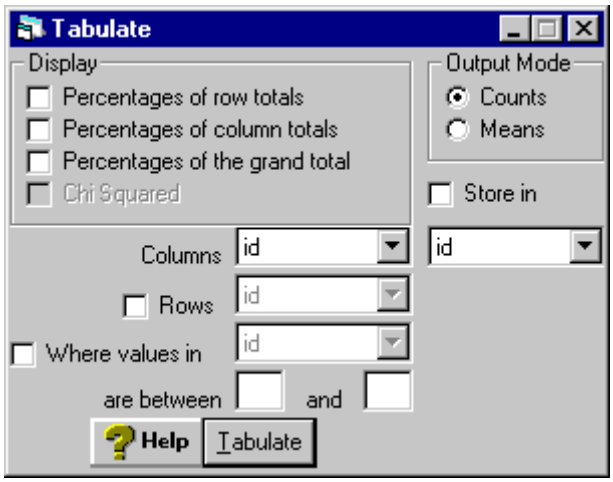
goto line 1 view Help Font

	reading(2442)	age(2442)	occasion(2442)	student(2442)
1	6.3009	-2.6504	1	1
2	6.8948	-1.6104	2	1
3	7.7685	-0.6204	3	1
4	8.8804	0.3396	4	1
5	10.082	1.3396	5	1
6	MISSING	MISSING	6	1
7	3.9854	-2.6404	1	2
8	5.49	-1.6204	2	2
9	6.0804	-0.6104	3	2
10	6.9285	0.2696	4	2
11	7.6031	1.2796	5	2
12	11.322	4.0096	6	2
13	4.2633	-2.4304	1	3
14	5.4474	-1.4504	2	3
15	6.8682	-0.4604	3	3
16	8.3667	0.4796	4	3
17	9.1808	1.4896	5	3
18	13.315	4.2096	6	3
19	4.2633	-2.5004	1	4

The data are now in the required form with one row per occasion. It would now be a good idea to save the worksheet, using a different name.

Initial data exploration

Before we start to do any modelling, we can do some exploratory work. The mean reading score at each occasion is obtained using the **tabulate** window available from the **Basic statistics** menu :



Select **means** as the **Output mode**

A drop down list labelled **variate column** appears. From the list select **reading.**

From the columns drop down list select **occasion**

Press **Tabulate**

Which produces the output :

```
Variable tabulated is reading
```

	1	2	3	4	5	6	TOTALS
N	277	371	356	294	262	198	1758
MEANS	4.72	5.69	6.67	7.62	8.61	11.3	7.13
SD'S	0.610	0.740	0.870	0.990	1.12	1.47	0.960

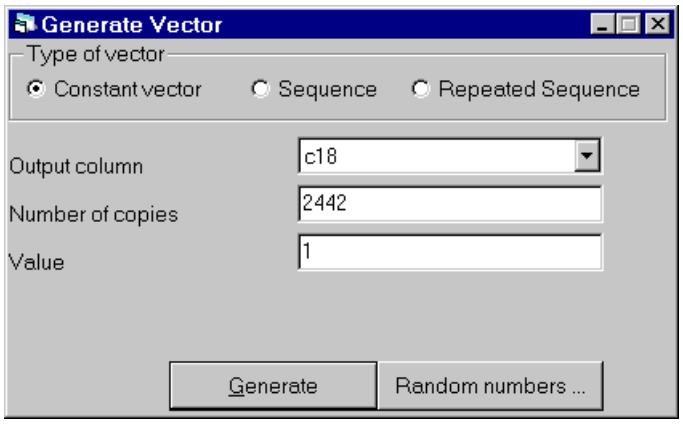
As we have noted earlier, our measure of reading is constructed from a series of different reading tests. The present scaling choice is reflected in the models which follow, where the increasing variance with age is modelled by fitting random coefficients.

Now use the tabulate window to tabulate mean age by occasion which produces :

```
Variable tabulated is age

            1         2         3        4        5       6    TOTALS

    N     277       371       356      294      262     198     1758

MEANS   -2.41     -1.44    -0.461    0.487     1.48    4.21  9.41e-06

 SD'S   0.145     0.154     0.157    0.127    0.131   0.115     0.142
```

The age variable has been transformed by measuring it as a deviation from the overall mean age. The mean reading score at each occasion is, from the way we have defined our reading scale, equal to the mean true age at that occasion, not the mean on the transformed age scale.

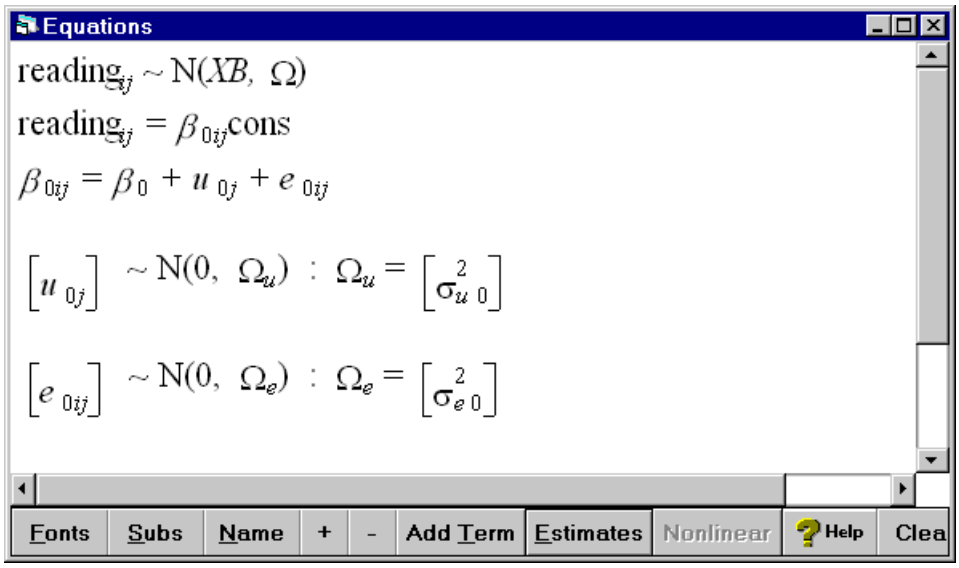
We are now almost in a position to set up a simple model but first we must define a constant column . We can do this by filling out the **generate vector** window available from the **data manipulation** menu as follows



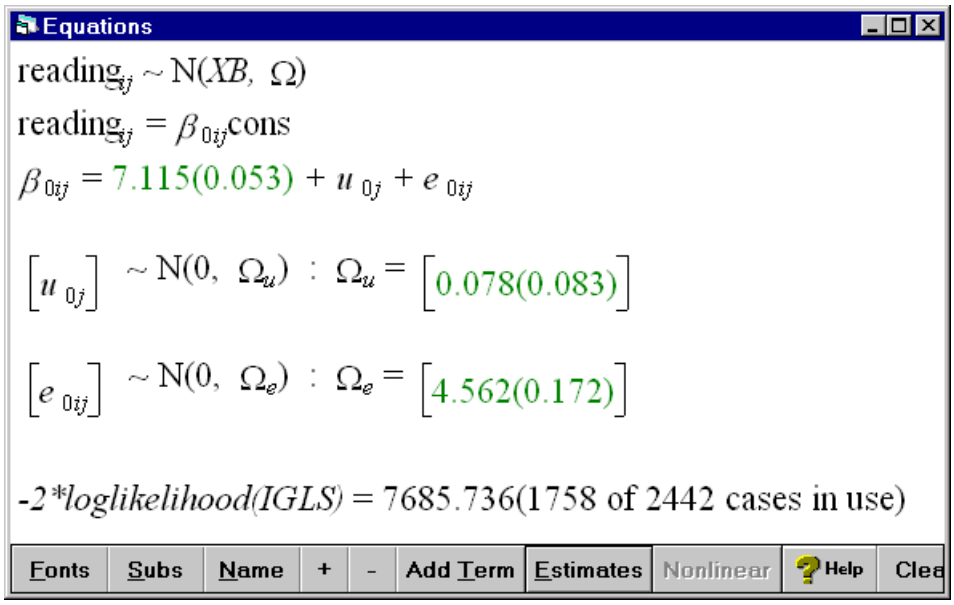
and clicking on **generate**. Use the **names** window to name c18 to be **CONS.**

A baseline variance components model

We start by seeing how the total variance is partitioned into two components: between students and between occasions within students. This variance components model is not interesting in itself but it provides a baseline with which to compare more complex models. We define the model and display it in the **equations** window as follows



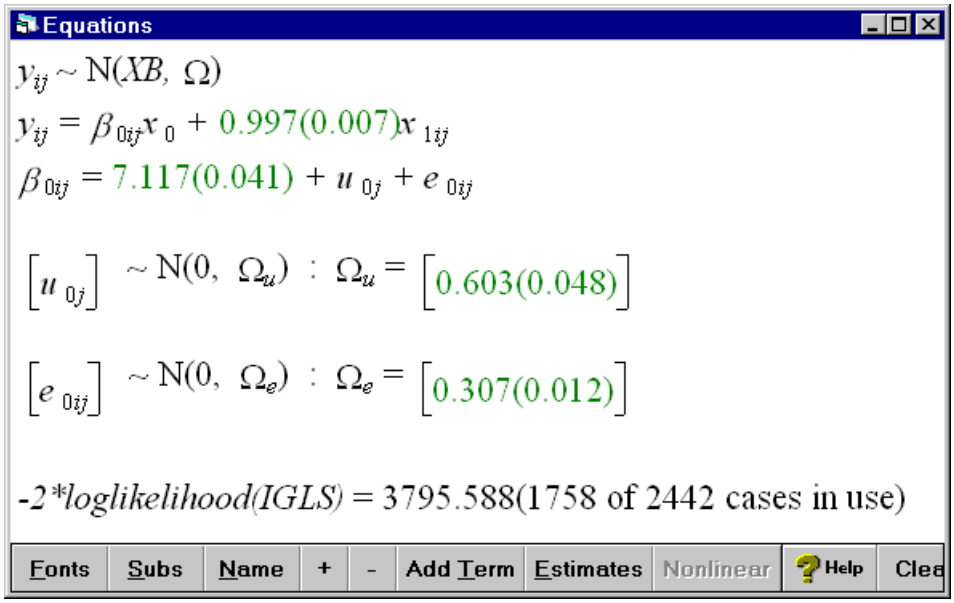
At convergence the estimates are:



As we would expect, given the way we have defined our response, the variation between occasions within students is large and overwhelms the variation between students. The likelihood statistic (-2 loglikelihood) is found at the bottom of the equation window and is here 7685.7: it can be used as the basis for judging more elaborate models.

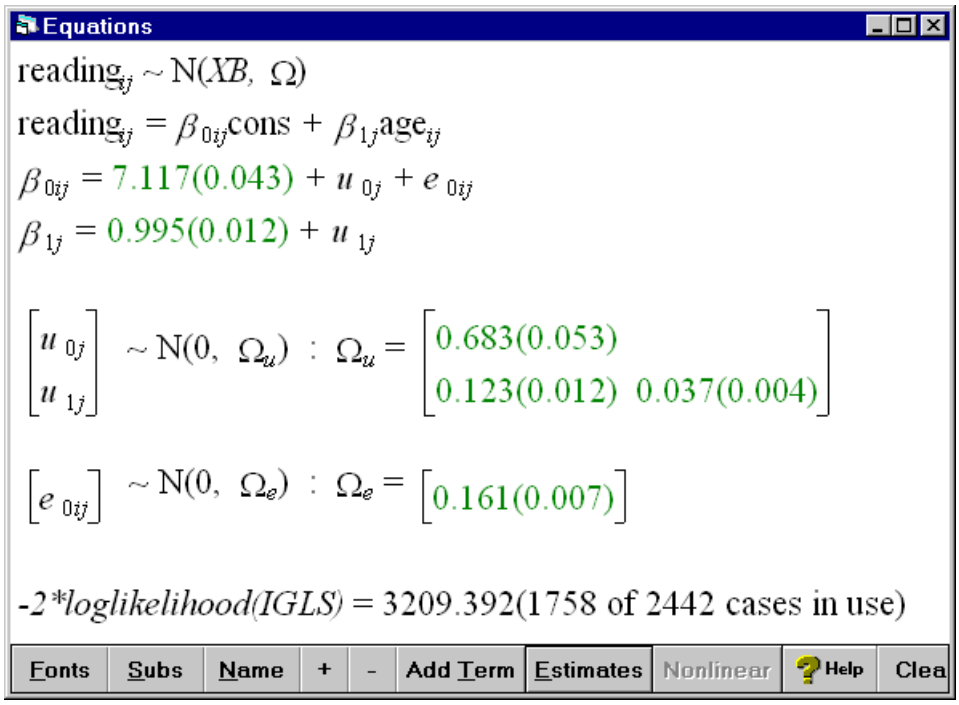
A linear growth curve model

A first step in modelling the between-occasion within-student, or level 1, variation would be to fit a fixed linear trend and so we add AGE to our list of fixed explanatory variables in the **equations** window, click on **more** and at convergence obtain



The estimate of the fixed parameter for AGE is very close to 1 because of the way the scale is defined. We see marked changes from our previous model in the estimates of the random parameters. We get an estimate of the level 2 variance which is about twice the size of the remaining level 1 variance, and a large reduction in the likelihood statistic, which is now 3795.6.

We would expect the linear growth rate to vary from student to student around its mean value of 1, rather than be fixed, and so we make the coefficient of age random at level 2 and continue iterations until convergence to give:

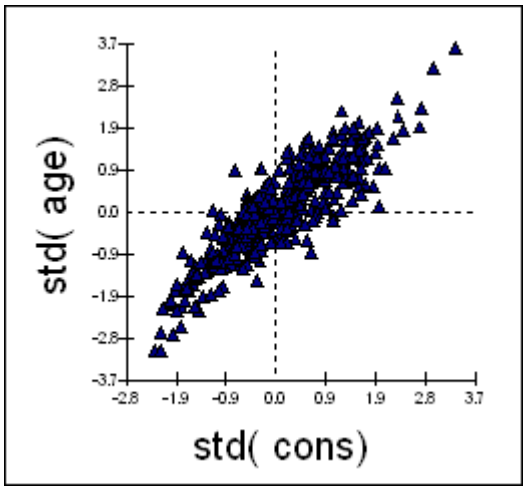


Note that the coefficient for AGE now has a subscript j.

The deviance, that is the reduction of 586 in the likelihood statistic is considerable and is clearly statistically highly significant. Hence there is considerable variation between students in their linear growth rates. We can get some idea of the size of this variation by taking the square root of the slope variance (σ_{u1}^2) to give the estimated standard deviation (0.19). Assuming Normality, about 95% of the students will have growth rates within two standard deviations of the overall mean (i.e. 1); this gives a 95% coverage interval of 0.62 to 1.38 for the `growth rate'.

We can also look at various plots of the level 2 residuals, using the **residuals** window. In Figure 10.2 we plot the level 2 standardised residuals

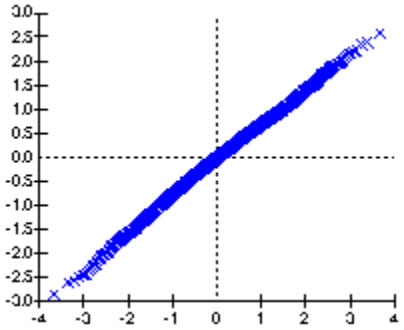
Figure 10.2. Level 2 standardised residual plot: slope Vs intercept



We see from the above window that the two level 2 residuals are positively correlated. Using the estimates window we see that the model estimate is 0.77 and shows that the greater the expected score at mean age, the faster the growth. However, this statistic needs to be interpreted with great caution: it can vary according to the scale adopted, and is relevant only for linear growth models.

To study the distributional assumptions we can plot the level 1 and level 2 residuals against their Normal scores: in the present case these plots conform closely to straight lines. The level 1 plot of the standardised residual against Normal score is as follows:

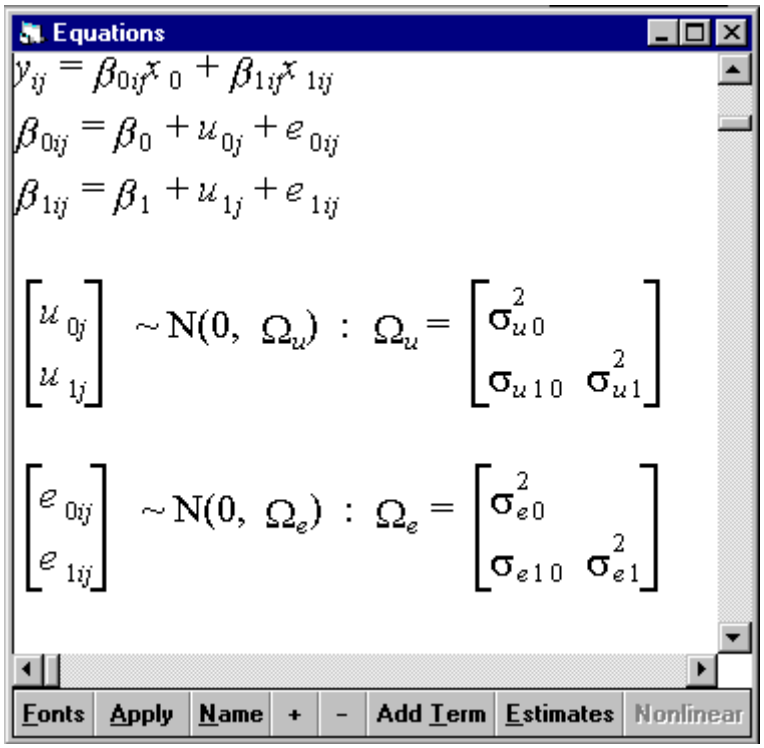
Figure 10.3. Level 1 standardised residual plot



Complex level 1 variation

Before going on to elaborate the level 2 variation we can model complex, that is non-constant, variation at level 1 to reflect the `constant coefficient of variation' scaling of the reading score. This requires that the total variance at each age is proportional to the

square of the mean, so that we would expect both the level 2 and level 1 variances to be non-constant. To allow the level 1 variance to be a quadratic function of the predicted value we declare the coefficient of age to be random at level 1 (see for example, Goldstein, 1995a, Chapter 3). The **equations** window is:



From the **variance function** window we see that the level 1 variance is the following function of the level 1 parameters, whose estimates are obtained by running the model to convergence:

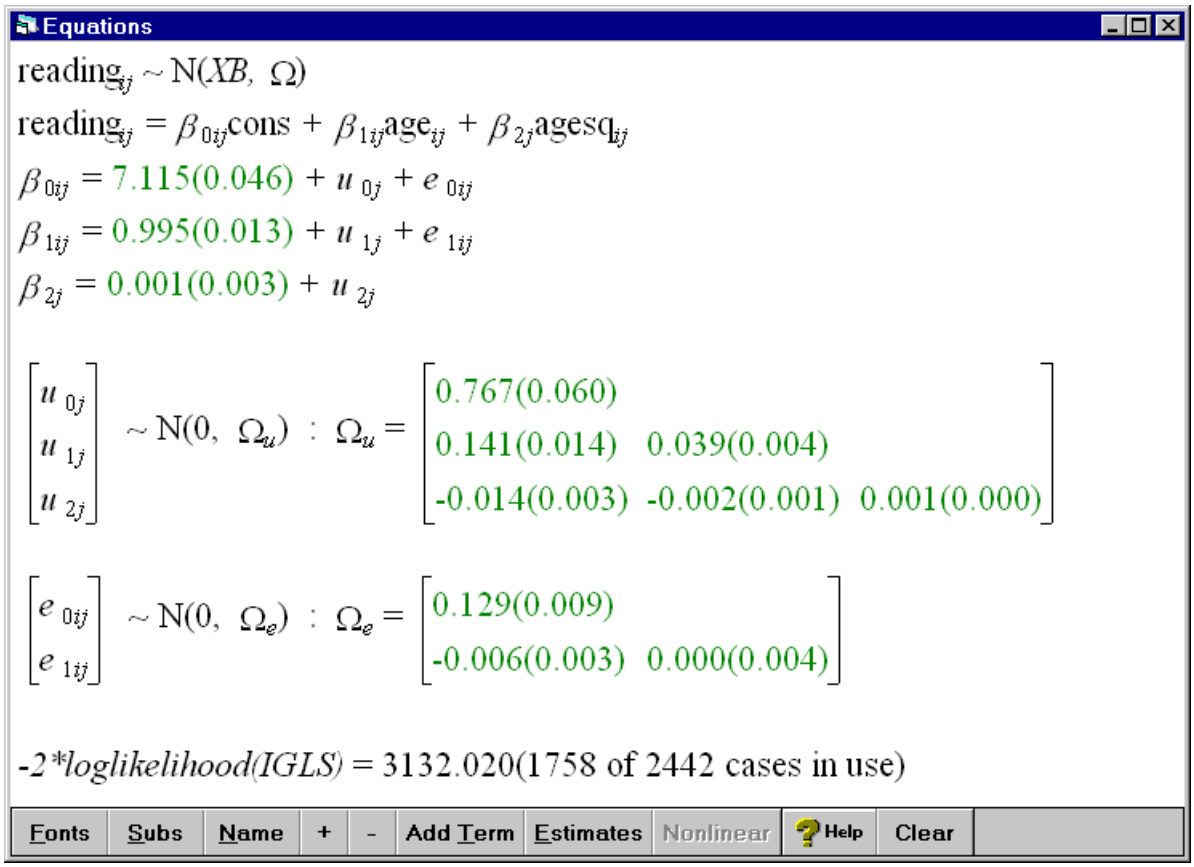
$$\text{var}(e_{0ij}\text{cons} + e_{1ij}\text{age}_{ij}) = \sigma^2_{e0}\text{cons}^2 + 2\sigma_{e01}\text{cons} * \text{age}_{ij} + \sigma^2_{e1}\text{age}_{ij}^2$$

There is a statistically significant decrease in the likelihood statistic of 32.4 with 2 degrees of freedom. We shall see later that some of this level 1 variation can be explained by further modelling of the level 2 variation.

Repeated measures modelling of non-linear polynomial growth

Growth in reading may not be linear for all students over this age range. One simple way of inducing non-linearity is to define a quadratic term in age. Use the **Calculate** window on the **Data manipulation** menu to calculate C19='age'^2 and then name C19

to be **agesq**. Add **agesq** to the model in the fixed part with a coefficient random at the student level. At convergence we have:

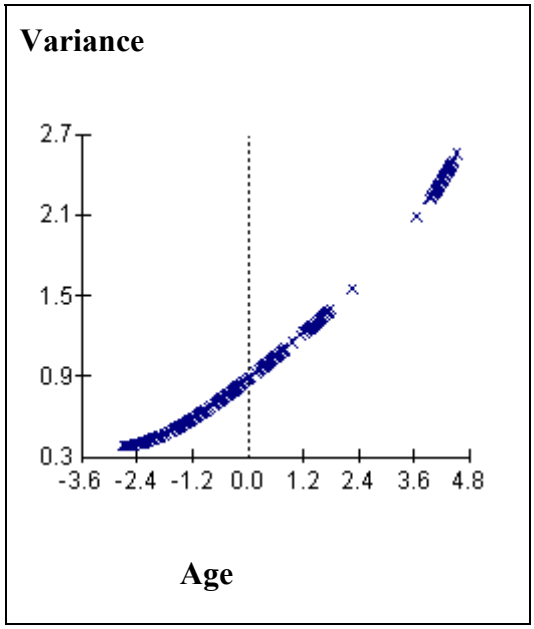


The likelihood statistic shows a further fall, this time of 45 with 4 degrees of freedom (one fixed parameter and three random parameters), and so there is strong evidence that a quadratic term which varies from student to student improves the model. Note that the fixed parameter AGESQD is very small because of the way the scale was defined over age. The level 1 random parameter estimates are different from those of the previous model. The σ_{e1}^2 parameter is extremely small and non-significant, and we estimate a linear effect with age for the between-occasion variance. The display precision in the above window is 3 significant digits after the decimal point. If this is increased to 4 using the **Display precision** item on the **Options** menu, we see that the estimate is actually 0.0004.

What has happened is that the more complex level 2 variation which we have introduced in order to model non-linear growth in individuals has absorbed much of the residual level 1 variation in the earlier model. We can view this final model for the random variation as a convenient and reasonably parsimonious description of how the overall variance produced by the assumption of a constant coefficient of variation is partitioned between the levels. We can use **the variance function** window to calculate the variance at both level 1 and level 2 for each record in the dataset. If we place these into separate columns (say, C28 and C29) and then add the two columns together into,

say, C30 using the *CALC* window, we obtain the total predicted variance and Figure 10.4 shows this as a function of age, confirming the original definition of the variance as a quadratic function of the mean, which itself is defined to be a linear function of age.

Figure 10.4. Total variance as a function of age.

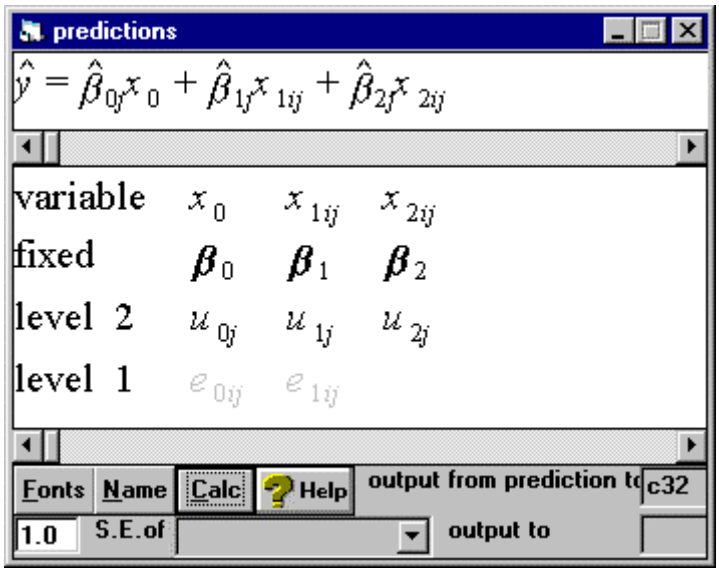


Since the overall relationship of the mean and variance with age is to some extent arbitrary and our choice, our principal interest lies in how the growth of individuals varies. The model parameters and derived variance functions describe these growth patterns and we can also display the estimated growth lines for selected individuals or groups. For example, to plot the lines for the first four individuals let us set up a *filter* column, say C31, which is 1 if the record belongs to one of these individuals and zero otherwise. This is achieved by typing in the **calculate window**:

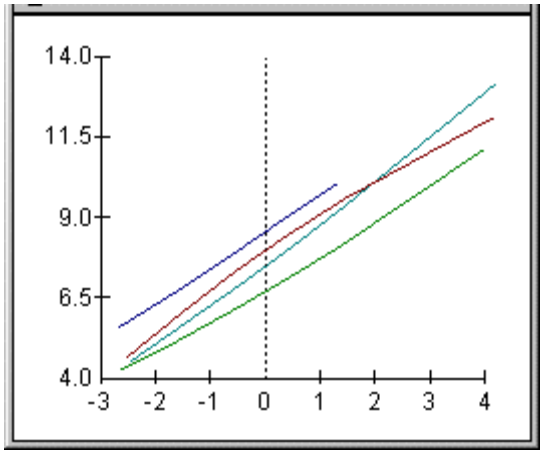
```
c31='student'<5
```

See the **help** system for a detailed description of how to use this window.

Now predict, using the **Predictions** option on the **Model** menu, from the fixed part plus level 2 random coefficients, into column 32, as follows:



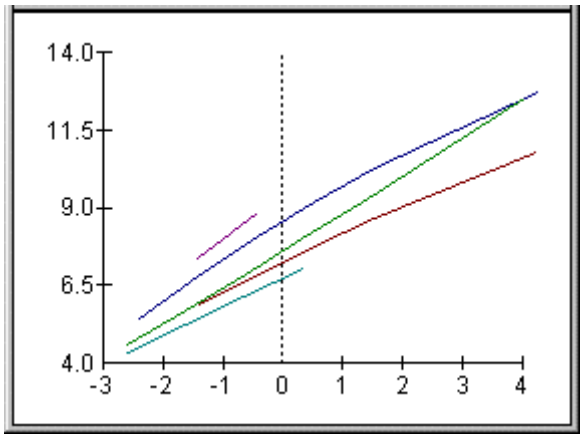
On the **Graph** menu plot C32 as Y, '*AGE*' as X. Choose C31 as the filter column and '*STUDENT*' as the group variable. Choose a line plot and in the **colour** box choose **rotate**. Click on **Apply** and the following plot will appear, with each student represented by a line.



We can easily display other student lines just by re-defining C31 with the following calculation

```
C31 = 'student'>=10 & 'student'<15
```

Which immediately updates the graph to display the predicted lines for students 11-15 as follows, where one student only has two measurements:



We can set up quite general filter functions using the **calculate window**, allowing extensive data exploration to take place.

Various extensions are available. We can fit multivariate repeated measures models using the **multivariate** model definition window as described in a later chapter, or extend the level 1 component to have a serial correlation structure (see Goldstein et al., 1994) using the time series macros described in the advanced multilevel modelling guide (Yang et al, 1999).

What you should have learnt from this chapter:

- How to formulate repeated measures models
- How to fit growth curve models of increasing complexity

Chapter 11: Multivariate response models

Multivariate response data are conveniently incorporated into a multilevel model by creating an extra lowest level which defines the multivariate structure. *MLwiN* has a special **Multivariate model** window for doing this.

We shall be using data consisting of scores on two components of a science examination taken in 1989 by 1905 students in 73 schools in England. The examination is the General Certificate of Secondary Education (GCSE) taken at the end of compulsory schooling, normally when students are 16 years of age. The first component is a traditional written question paper (marked out of a total score of 160) and the second consists of coursework (marked out of a total score of 108), including projects undertaken during the course and marked by each student's own teacher but 'moderated', i.e. a sample checked, by external examiners. In the worksheet both scores have been scaled so that the maximum is 100. Interest in these data centres on the relationship between the component marks at both the school and student level, whether there are gender differences in this relationship and whether the variability differs for the two components.

This chapter will show how to specify and fit a relatively straightforward multivariate model with Normal responses. We shall not deal with the case of categorical responses: this case, with ordered or unordered categories, is discussed in a separate manual supplied with *MLwiN* (Yang et al, 1999). We shall mention some extensions of the basic model in the following sections.

Specifying a multivariate model

To define a multivariate, in the case of our example a 2-variate, model we treat the individual student as a level 2 unit and the 'within-student' measurements as level 1 units. Each level 1 measurement 'record' has a response, which is either the written paper score or the coursework score. The basic explanatory variables are a set of dummy variables that indicate which response variable is present. Further explanatory variables are defined by multiplying these dummy variables by individual level explanatory variables, for example gender. Omitting school identification, the data matrix for three students, two of whom have both measurements and the third who has only the written paper score, is displayed in Table 4.1. The first and third students are female (1) and the second is male (0).

Table 4.1 Data matrix for examination data.

		Intercepts		Gender	
Student	Response	Written	Coursework	Written	Coursework
1 (female)	y_{11}	1	0	1	0
1	y_{21}	0	1	0	1
2 (male)	y_{12}	1	0	0	0
2	y_{22}	0	1	0	0
3 (female)	y_{13}	1	0	1	0

The statistical model for the two level model ignoring school, is written as follows:

$$y_{ij} = \beta_0 z_{1ij} + \beta_1 z_{2ij} + \beta_2 z_{1ij} x_j + \beta_3 z_{2ij} x_j + u_{1j} z_{1ij} + u_{2j} z_{2ij}$$

$$z_{1ij} = \left\{ \begin{array}{c} 1 \text{ if written} \\ 0 \text{ if coursework} \end{array} \right\}, \quad z_{2ij} = 1 - z_{1ij}, \quad x_j = \left\{ \begin{array}{c} 1 \text{ if female} \\ 0 \text{ if male} \end{array} \right\}$$

$$\text{var}(u_{1j}) = \sigma^2_{u1}, \quad \text{var}(u_{2j}) = \sigma^2_{u2}, \quad \text{cov}(u_{1j}, u_{2j}) = \sigma_{u12}$$

There are several interesting features of this model. There is no level 1 variation specified because level 1 exists solely to define the multivariate structure. The level 2 variances and covariance are the (residual) between-student variances. In the case where only the intercept dummy variables are fitted, and in the case where every student has both scores, the model estimates of these parameters become the usual between-student estimates of the variances and covariance. The multilevel estimates are statistically efficient even where some responses are missing, and in the case where the measurements have a multivariate Normal distribution IGLS provides maximum likelihood estimates.

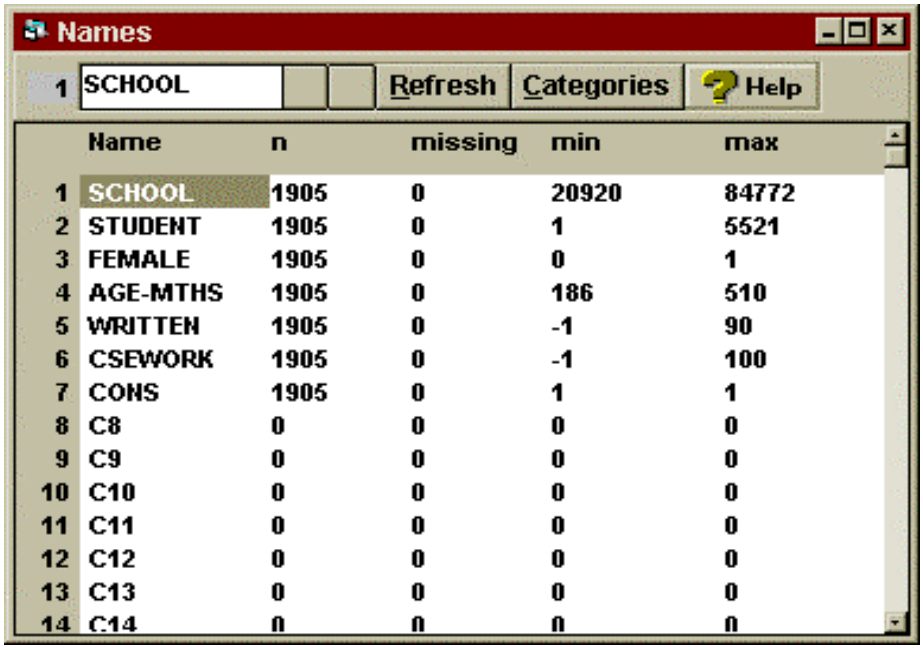
Thus, the formulation as a 2-level model allows for the efficient estimation of a covariance matrix with missing responses, where the missingness is at random. This means, in particular, that studies can be designed in such a way that not every individual has every measurement, with measurements randomly allocated to individuals. Such 'rotation' or 'matrix' designs are common in many areas and may be efficiently modelled in this way. A more detailed discussion is given by Goldstein (1995a, Chapter 4) and the ability to provide estimates of covariance matrices at each

higher level of a data hierarchy enables further models such as multilevel factor analyses to be fitted (see Rowe and Hill, 1997).

A third, school, level can be incorporated and this is specified by inserting a third subscript, *k*, and two associated random intercept terms.

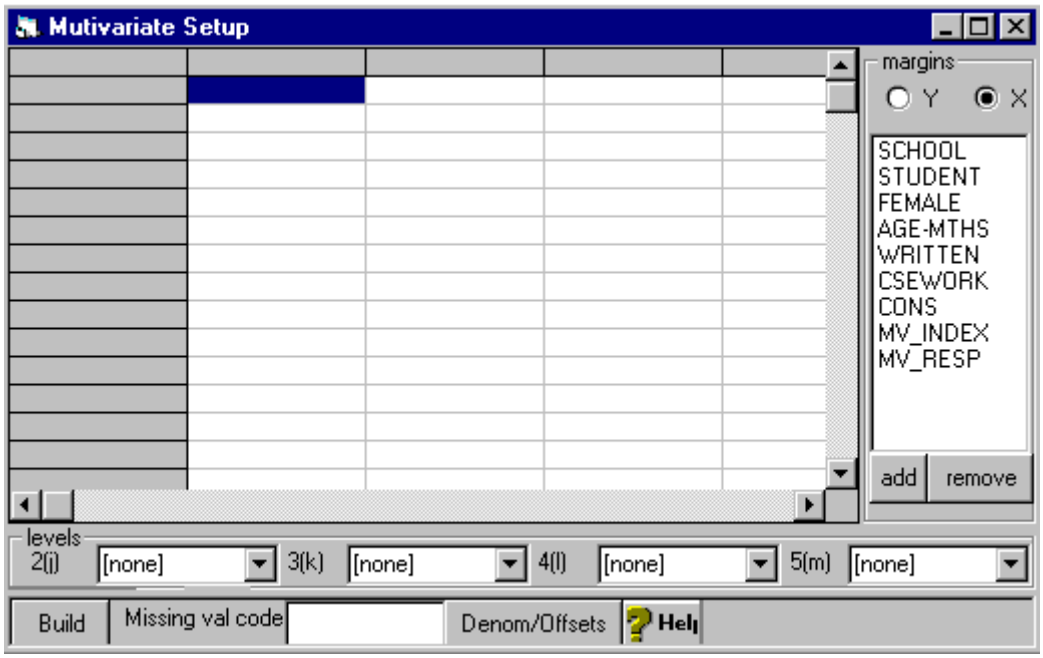
Reading in the data and setting up the model

Retrieve the worksheet GCSEMV.ws, supplied with the *MLwiN* software, and open the **names** window from the **Data manipulation** menu as follows



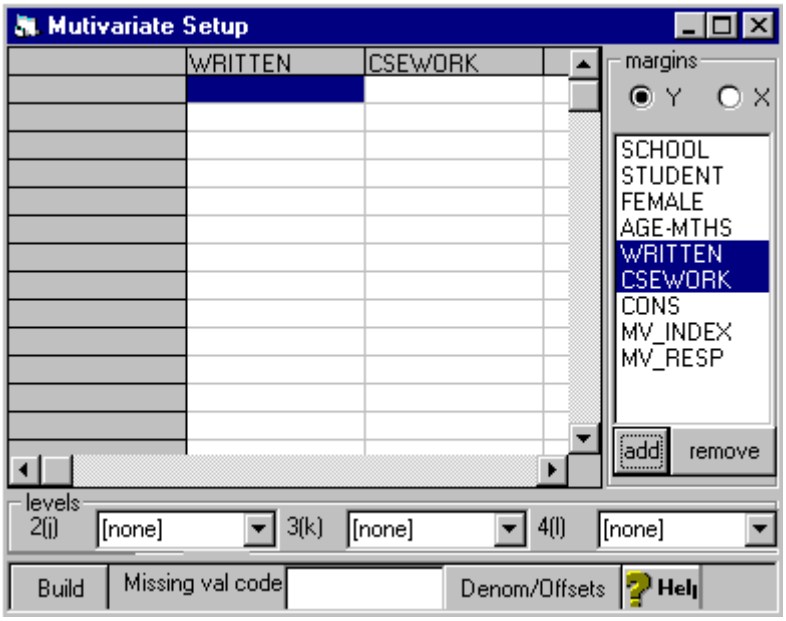
	Name	n	missing	min	max
1	SCHOOL	1905	0	20920	84772
2	STUDENT	1905	0	1	5521
3	FEMALE	1905	0	0	1
4	AGE-MTHS	1905	0	186	510
5	WRITTEN	1905	0	-1	90
6	CSEWORK	1905	0	-1	100
7	CONS	1905	0	1	1
8	C8	0	0	0	0
9	C9	0	0	0	0
10	C10	0	0	0	0
11	C11	0	0	0	0
12	C12	0	0	0	0
13	C13	0	0	0	0
14	C14	0	0	0	0

The two response variables each have approximately 10% missing, so that about 20% of students have a single response: the missing values are coded -1. For present purposes we assume that missing is completely at random. The data above have a single record for each student, whereas we require a record for each response. Thus the data need to be re-organised and we do this via the **multivariate model** window. Click on **Model/Multivariate** and the following appears:

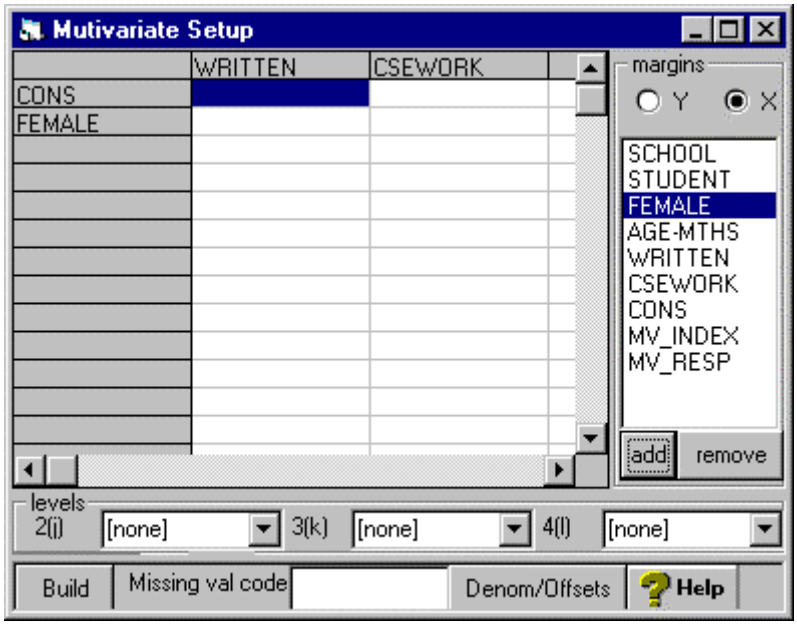


A list of variables appears and we have the options of selecting those which are responses and those which are predictor or explanatory variables. Note that two variables have been added to the list 'mv_index' and 'mv_resp'. These are generated automatically on opening this window; they refer respectively to the (level 1) identification of the response variable and the new response variable formed by stacking the responses into a single column.

So first click on 'written' and 'csework' (hold down the control key for multiple selections), click on **Y** and then **add** to give the following



Along the top the response variables are listed and for each of these a corresponding explanatory variable needs to be created. To do this click on **X** and then select the explanatory variables. In the present analysis we are going to use **cons,** and **female** (female=1); so in turn select these and click **add.** These two explanatory variables will be listed in the first two rows of the table. We obtain the following screen

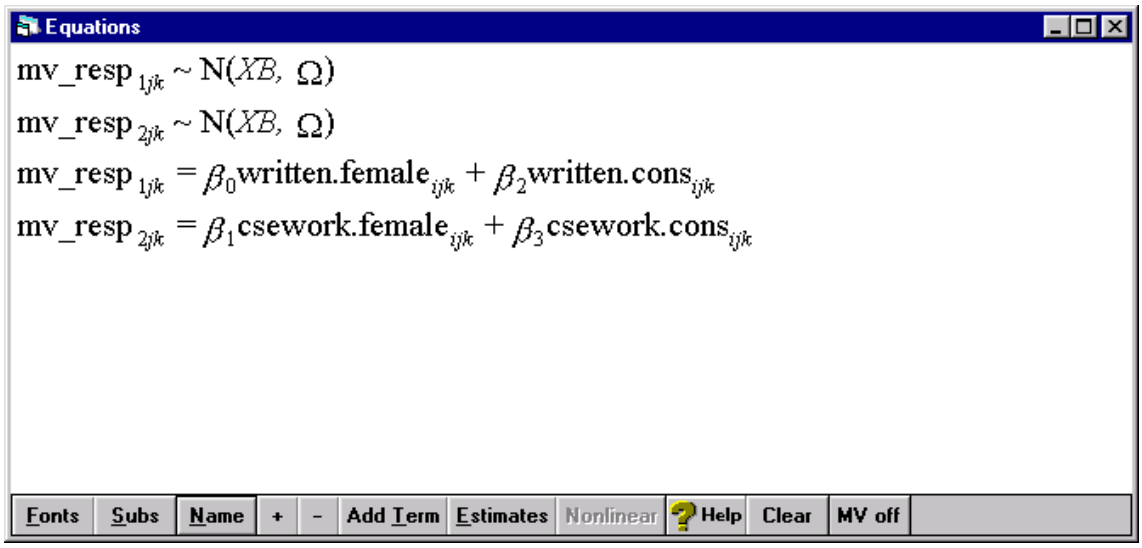


Clicking in each of the four cells defined so far, *MLwiN* selects the first character of each response variable and the response sequence number (reading from left to right) and combines it with the first four characters of each explanatory variable, to form a new name. You can subsequently change these names if you wish. If a variable is deleted this does not cause the remaining names to change. Note also that all these new names are assigned columns at the end of the worksheet; in the present case since there are 400 columns, C395-C400 have been assigned so far. We now have defined all the variables as required by the structure given in Table 4.1. Note that the explanatory variables defined in this way are entered into the model in the order in which they are displayed in the window, from left to right and from top to bottom.

Finally we need to insert the level identifiers (level 2 is student and 3 is school) and the missing value code of -1. Having done this click on **Build** for *MLwiN* to create the data and the basic model structure. In the next section we shall start by looking at what has so far been created in the equations window, and then finish our model specification.

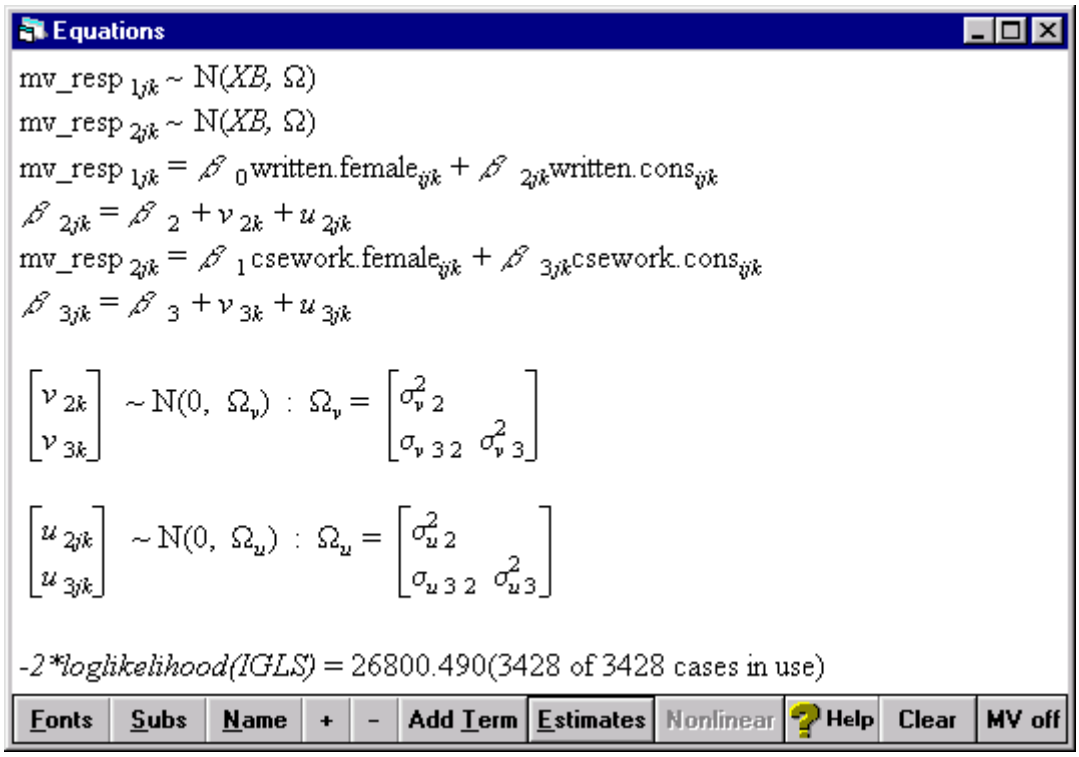
Using the equations window to complete the model setup

Having set up the basic data structure in the multivariate window, and clicked on **Build**, open up the equation window with the following display, showing the variable names:

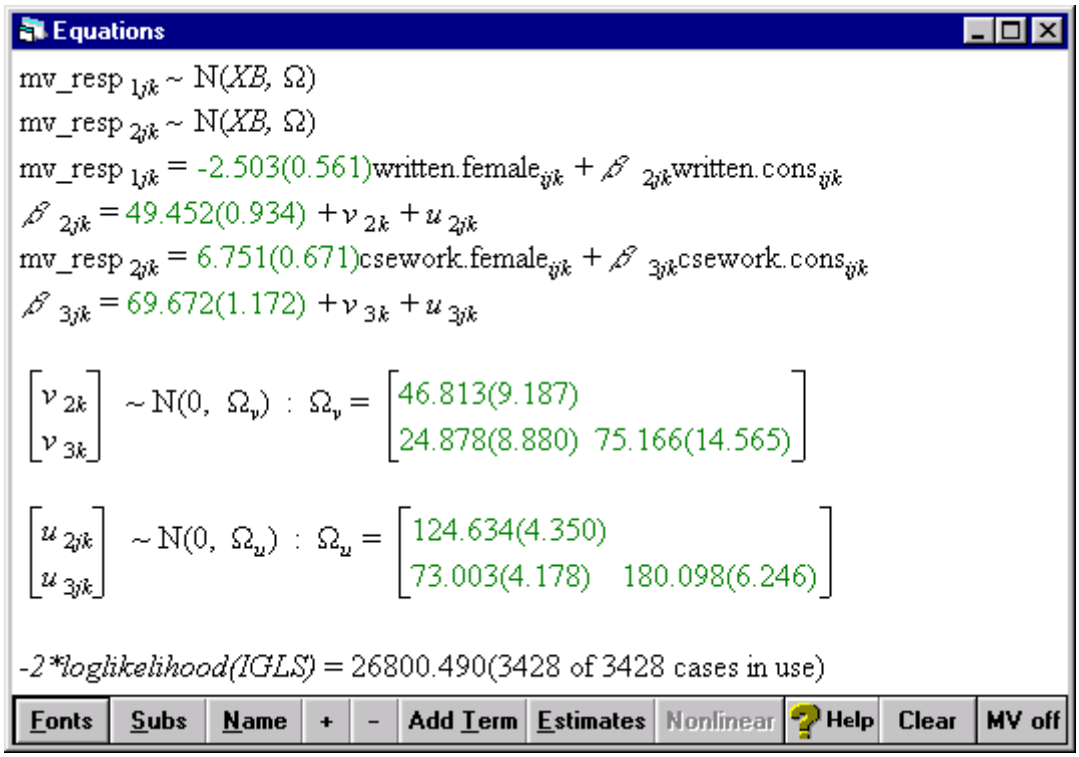


The response is called by the default name **mv_resp** where the first subscript (1 or 2) denotes the response variable, **written** or **coursework**. Each of the responses is a linear function with an intercept and gender term. Thus, in the fixed part of the model we will estimate an intercept and gender coefficient for each response. Note that the order in which the explanatory variables occurs is determined by the order in which they ere entered in the multivariate setup window.

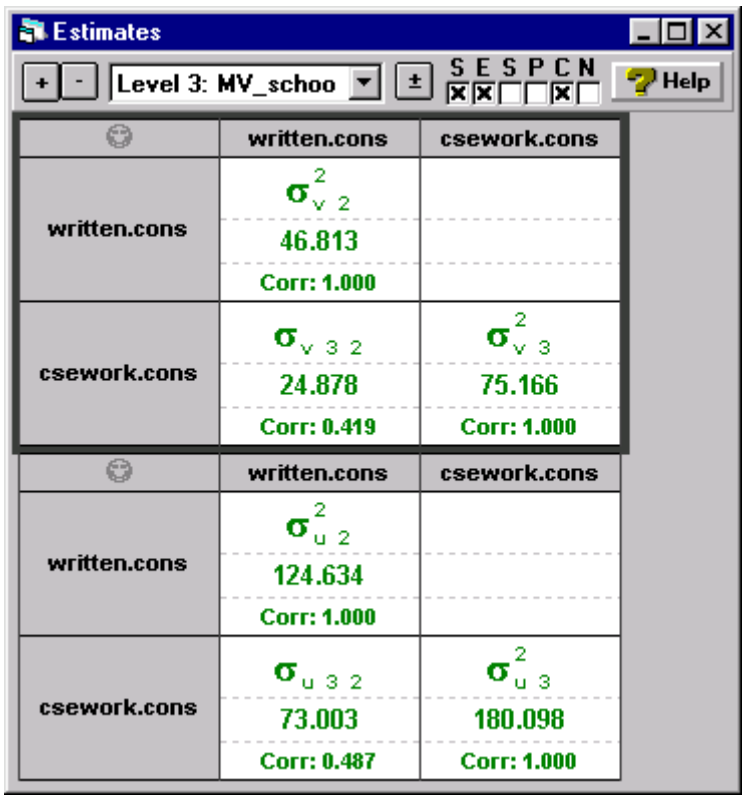
We will now specify the random part of the model, fitting the (residual) 2 x 2 covariance matrix for the responses at both student and school level. To do this simply click on the two intercept coefficients, β_2, β_3, and specify them to be random at student and school level. Expanding the model (click on +) we obtain the following full model specification:



Running the model to convergence we obtain the following parameter estimates:



The girls do somewhat worse on the written paper (-2.5) but considerably better than the boys on the coursework component of the examination (6.8). The coursework component also has a larger variance at both student and school level with correlations between coursework and written 0.42 and 0.49 at school and student level respectively. The intra-school correlation is 0.27 for the written paper and 0.29 for the coursework. We can often view the results more conveniently using the **Model/Estimate tables** window as follows (selecting **Numbers/Options** with 2 decimal places) to show just the estimates and correlations at level 3 and level 2.



A more elaborate model

We now add gender ('female') to the model as a random coefficient at the school level. We need to click on both the displayed coefficients of this variable to do this. Running the model to convergence gives the following results for the school level covariance matrix, where the order of the random terms is the intercept for the written and coursework, followed by the gender difference for written and coursework.

$$\Omega_v = \begin{bmatrix} 54.548(11.774) & & & \\ 38.893(12.686) & 104.694(21.827) & & \\ -7.458(5.688) & -6.975(7.299) & 5.289(4.244) & \\ -21.107(9.967) & -39.674(14.696) & 11.005(6.3) & 49.908(14.606) \end{bmatrix}$$

with a likelihood statistic value of 26756.1, so that the deviance statistic is 44.4 with 7 degrees of freedom, which is highly significant. We notice, however, that the estimate for the variance of the gender difference for the written paper is close to its standard error, as are the covariances with the gender difference. Fitting the model with this random term omitted gives the following results at school level

$$\Omega_v = \begin{bmatrix} 47.149(9.243) & & \\ 33.003(10.79) & 101.199(20.848) & \\ -11.699(7.638) & -34.028(12.827) & 40.483(11.868) \end{bmatrix}$$

with a likelihood statistic of 26760.4 so that, compared with the preceding model the deviance is only 4.3 with four degrees of freedom. Thus, while there is variation among schools in the gender difference for the coursework component, there is no evidence that there is any substantial difference for the written paper. From these estimates we can compute, for example, the school level correlation between male and female coursework scores which is 0.78. This value is computed from the above table as follows.

Denoting the school level random variable associated with coursework for males by *m* and that for females by *f*, since the gender coefficients represent the males effect and the male – female difference we have:

$$\text{var}(m) = 101.2; \quad \text{var}(f-m) = 40.5; \quad \text{cov}[m(f-m)] = -34.0$$

from which we obtain cov(*mf*)=67.2, var(*f*)=73.7 and the above correlation is computed as $67.2/\sqrt{73.7\times 101.2} = 0.78$. Although this is rather high it may be of interest to compute separate estimates for males and females for each school. To do this we go to the **Model/Residuals** window and calculate the level 3 school residuals. Since there are 3 random coefficients at this level (the intercepts for written and coursework and the girl-boy difference coefficient for coursework), these, by default will be placed into C300-C302. Thus C301 contains the school level intercept residual for boys on the coursework paper and C302 the girl-boy difference for the coursework paper. If we calculate

```
c350=c302+c301
```

then C350 will contain the school level intercept residual for girls coursework. Plotting this against that for the boys (C301) shows that there is one school in particular (with ID=64343) which appears to do particularly well for its girls but only at about the average for boys. By contrast, the school with ID=20920 performs at the mean for its girls but well below the mean for boys. Both these schools might be interesting to follow up.

Although we do not pursue it here, we could go on to fit complex student level variation, with different variances for boys and girls etc. We can also go on to make predictions from the fixed part of the model, or for each school for each response, using the **Model**/**Predictions** window. This would enable us, for example, to see which schools had particularly large values of the gender difference in the coursework component. Since the multivariate model is specified simply by adding another lowest level we can combine a multivariate model specification with other types of multilevel model such as repeated measures data (Chapter 3) or cross classifications (see Appendix A).

Note that if you wish to run a single response model following a multivariate model click on the **MV OFF** button along the bottom toolbar of the **equation** window which appears when a multivariate model is specified.

What you will have learnt from this chapter

- An understanding of multivariate multilevel models
- How to specify a multivariate model in *MLwiN*

Chapter 12: Diagnostics for Multilevel Models

Diagnostic procedures, such as the detection of outliers and data points with a large influence on the fit of a model, are an important part of ordinary least squares regression analysis. The aim of this chapter is to demonstrate, via an educational example, how some of the concepts and diagnostic tools used in regression analyses can be translated into the multilevel modelling situation. The statistical techniques used and the example explored in this chapter are dealt with in detail in Langford and Lewis (1998).

Data exploration techniques, including the detection of outlying observations, are a little explored area of multilevel modelling. For ordinary regression, there is an extensive literature on the detection and treatment of single outliers, and an increasing literature on multiple outliers (Barnett and Lewis, 1994). However, in data structures of increasing complexity the concept of an outlier becomes less clear cut. For example, in a multilevel model structure, we may wish to know at what level(s) a particular response is outlying, and in respect of which explanatory variable(s). We use the term *level* to describe the unit of analysis in our model. In a multilevel model, more than one unit of analysis is appropriate for the data (Goldstein, 1995). In a simple educational example, we may have data on examination results in a 2-level structure with students nested within schools, and either students or schools may be considered as being outliers at their respective levels in the model. Suppose, for example, that at school level a particular school is found under test to be a discordant outlier; we will need to ascertain whether it is discordant due to a systematic difference affecting all the students measured within that school, or because one or two students are responsible for the discrepancy. At student level, an individual may be outlying with respect to the overall relationships found across all schools, or be unusual only in the context of his/her particular school. Indeed, these concepts become more complex when there are more than two levels.

Other concepts become similarly more complex. For example, masking of outlying observations, where the presence of one observation may conceal the importance of another (Atkinson, 1986), can apply across the levels of a model. The outlying nature of a school may be masked because of the presence of another similarly outlying school, or by the presence of a small number of students within the school whose influence brings the overall relationships within that school closer to the average for all schools. Other effects such as swamping (Barnett and Lewis, 1994) and other measures of joint or conditional influence (Lawrance, 1995) may also occur within as well as between units at the higher levels of a multilevel model.

An educational example

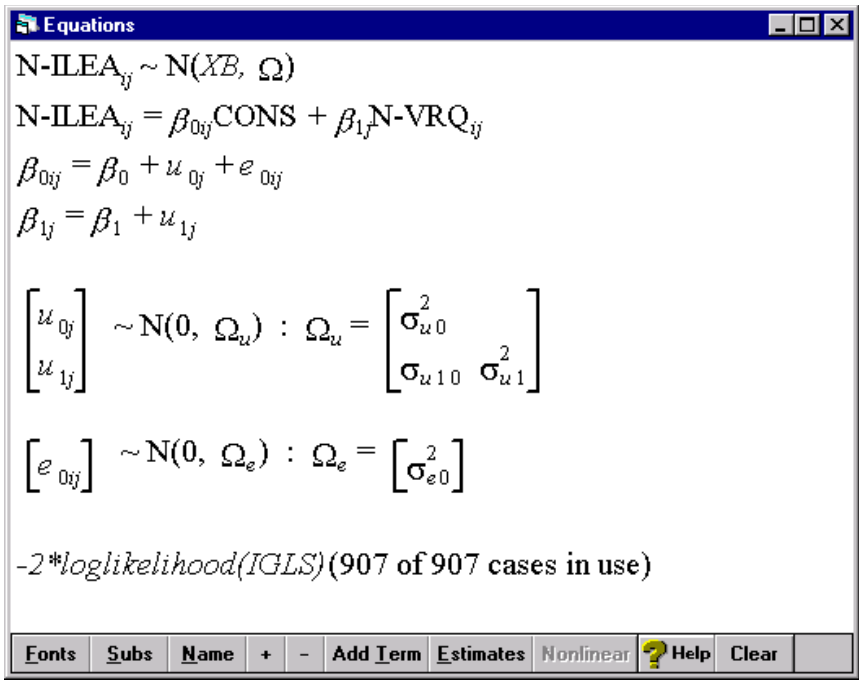
In their classic paper, Aitkin and Longford (1986) (see Goldstein (1995), p. ix) report an analysis of 907 students in 18 schools in a Local Education Authority in the United

Kingdom, discussing the implications of fitting different models to the data on parameter estimates and their interpretation. Of particular interest here is the presence of two single-sex grammar schools in the data which are otherwise made up of comprehensive schools. Our analysis focuses on whether these two schools are discordant outliers in the data set, and thus of a genuinely different character, More generally it uses the data as an example of how an examination of outliers in a two-level model may be pursued, issues not investigated by Aitkin and Longford. The numbers of students per school are given in table below, with school 17 and 18 being the grammar schools.

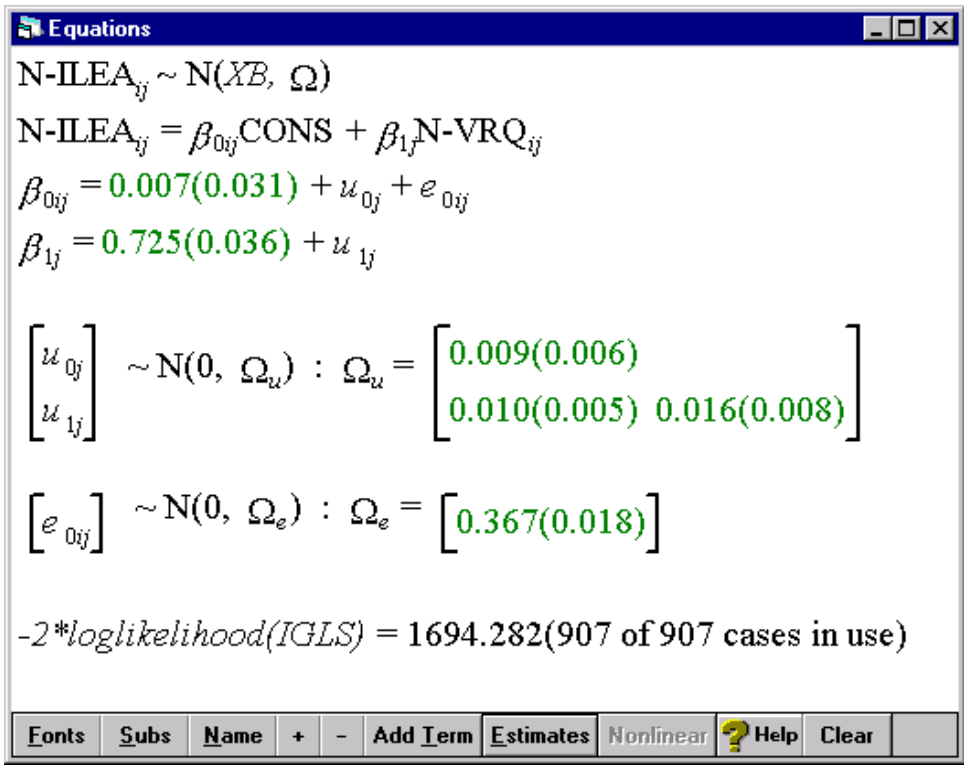
School	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
# Pupils	65	79	48	47	66	41	52	67	49	47	50	41	49	29	72	62	22	21

The outcome variable for the following analysis is the O-level/CSE examination results, converted into a score by adding up the results for individual subjects for each student using a simple scoring system. The "intake" score for each school is defined as being the Verbal Reasoning quotient, VRQ, a measure of students' ability made when they enter the school. Both of these scores were converted into Normal scores for this analysis, as there was evidence of clustering of scores at high values, probably determined by the fact that there is an upper limit on the scores which an individual student can achieve, hence our analysis is not exactly equivalent with that of Aitkin and Longford. The outcome variable for each pupil is referred to as N-ILEA and the intake variable, measuring VRQ, as N-VRQ.

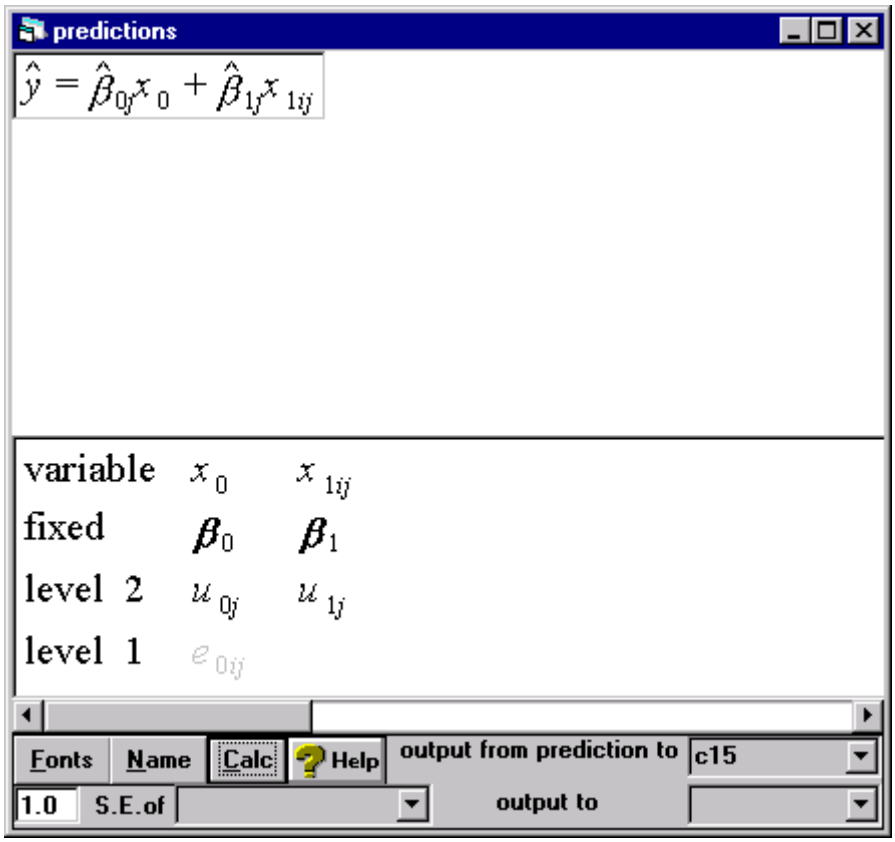
Open the worksheet called **diag1.ws** containing the data, which you can look at using the **Names** and **Data** windows. Go to the equation editor, click on **Names**, and double click on the + button to expand the model which is already set up in the worksheet so you have the following:



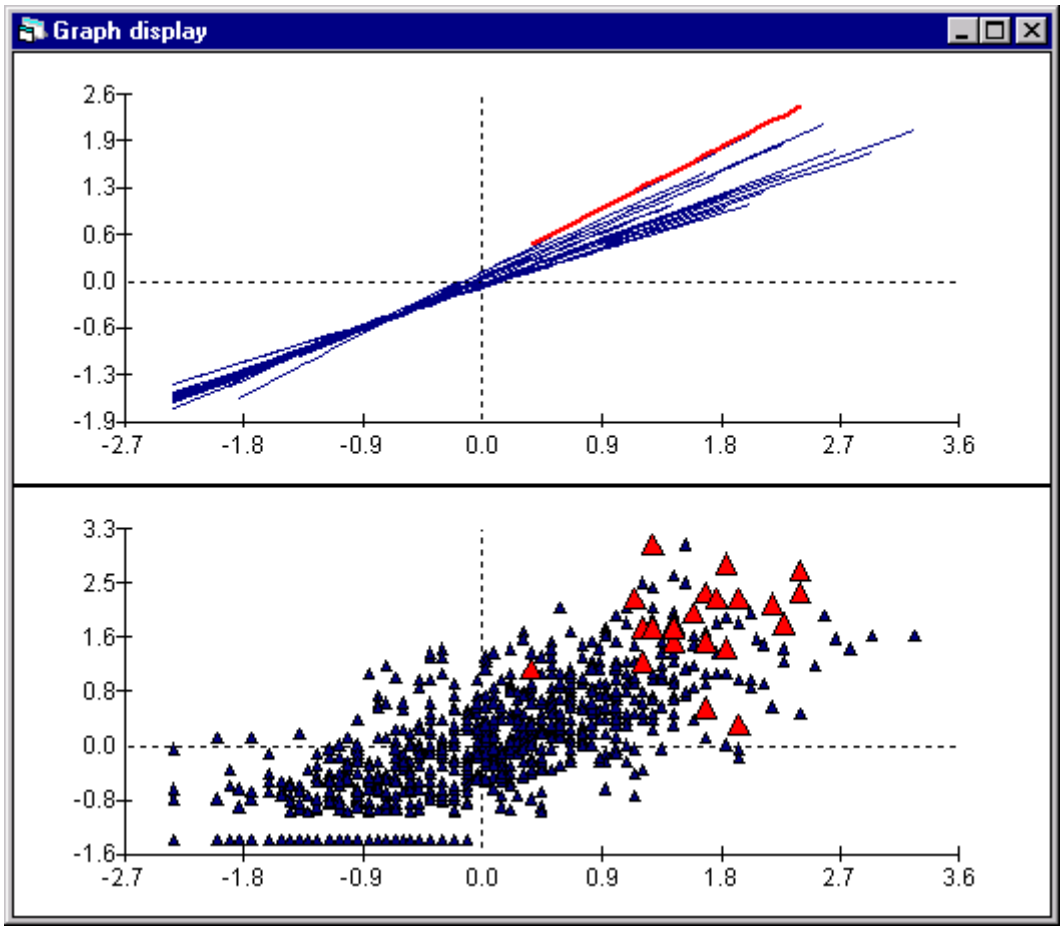
We have a two level model, with students (subscript *i*) nested within the 18 schools (subscript *j*). The outcome variable, N-ILEA is modelled as a function of the intake variable, N-VRQ, which is also in the random part of the model at level 2, the school level. This means we have a random slopes and intercepts model for schools. There is just one variance term at level 1 measuring the residual variance of the students. Double click on the **Estimates** button at the bottom of the **Equations** window, and run the model until it converges. The screen should look like this:



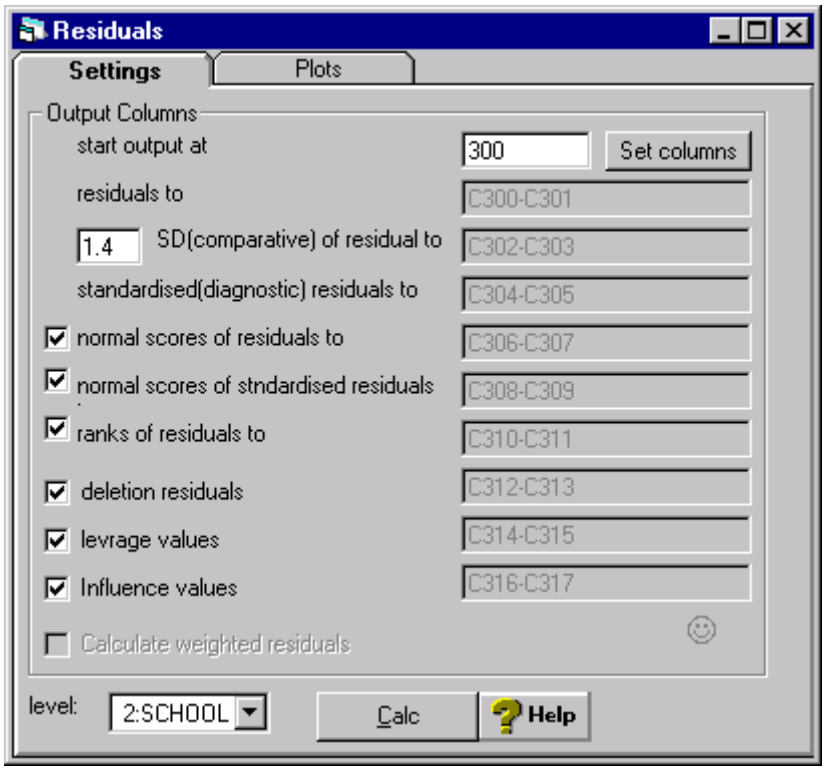
Most of the variance is occurring between students, but there is significant variance between schools at level 2. We can explore the relationship between outcome and intake variables for each school by using the **Predictions** window. Choose the fixed parameters and random parameters at level 2 to make predictions for the regression lines for each school, and save them into c15. The **Predictions** window should look like this when you make the calculation:



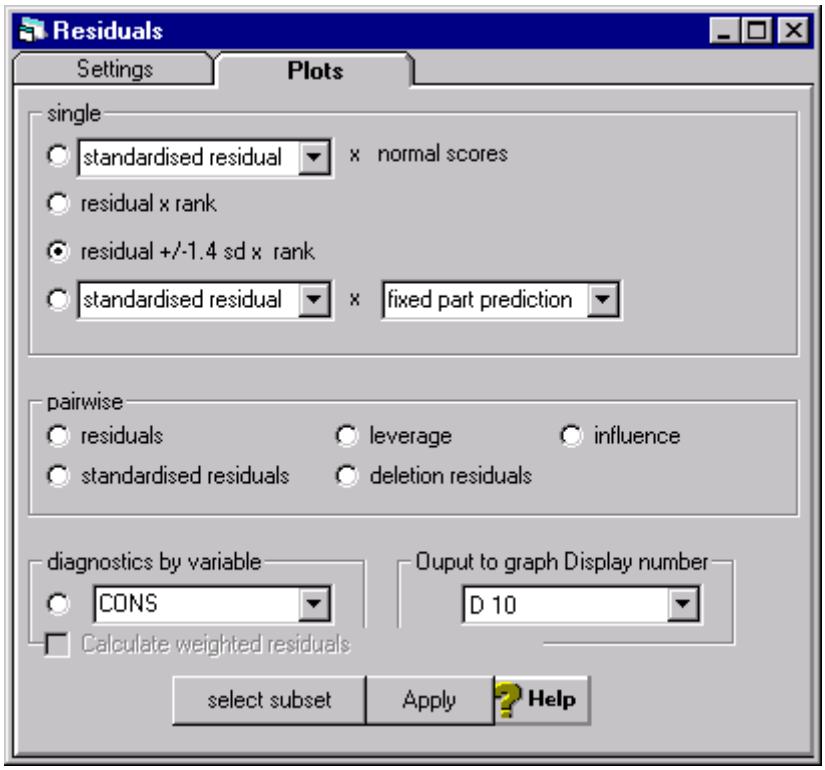
Use the **Graph** menu to create two graphs on the same page in graph **D1**: the first a line graph of the predictions in c15 plotted against N-VRQ, grouped by school; the second, a point plot of the response variable N-ILEA against N-VRQ, showing the outcome and intake scores for each student. Highlight the uppermost line on the graph by clicking just above it and choosing the highlighting options from the menu – this should be school 17, one of the grammar schools which we will focus this example analysis on. The graph display window should look like this, with school 17 and its pupils highlighted in red:



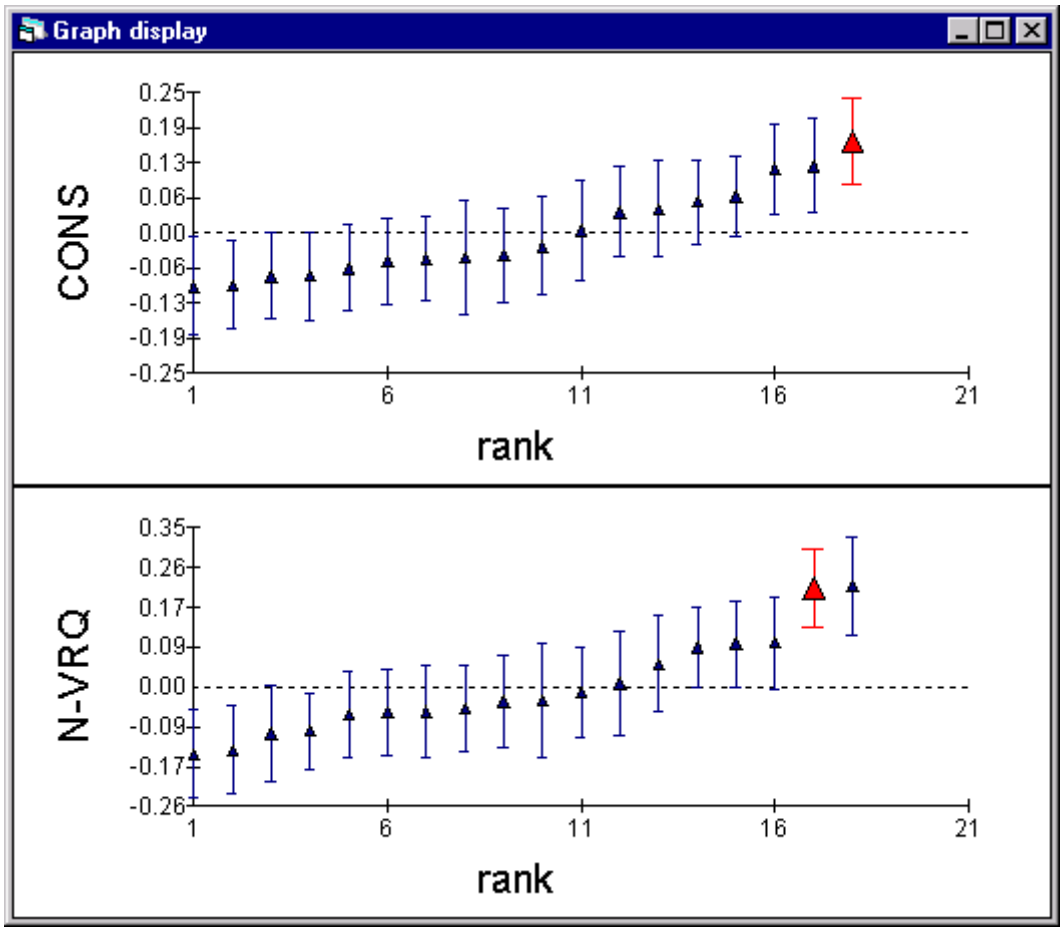
We can see that school 17 has the highest intercept value and one of the highest slope values. From the lower plot, we can also see that students in school 17 tend to be high achievers, with the possible exception of one student whose score is near the mean. We can examine the values of slopes and intercepts further by using the **Residuals** menu. Choose to calculate residuals and other regression diagnostics at level two from the **Residual Settings** menu, and calculate the results in to c300 onwards. Type the value 1.4 into **the SD (comparative) of residual** box so that we can compare confidence intervals around the residuals for each school. Goldstein and Healy (1995) discuss the circumstances where the value of 1.4 rather than the conventional 1.96 standard deviations is used to calculate 95% intervals. Roughly speaking, if we wish to use the intervals to make comparisons between pairs of schools then we can judge significance at the 5% level by whether or not the (1.4) intervals overlap. If, on the other hand, we wish, say, to decide whether a school is significantly different from the overall mean the conventional (1.96) interval can be used in terms of whether or not it overlaps the zero line. For present purposes we shall assume that interest focuses on pairwise school comparisons. The residual screen should now look like this:



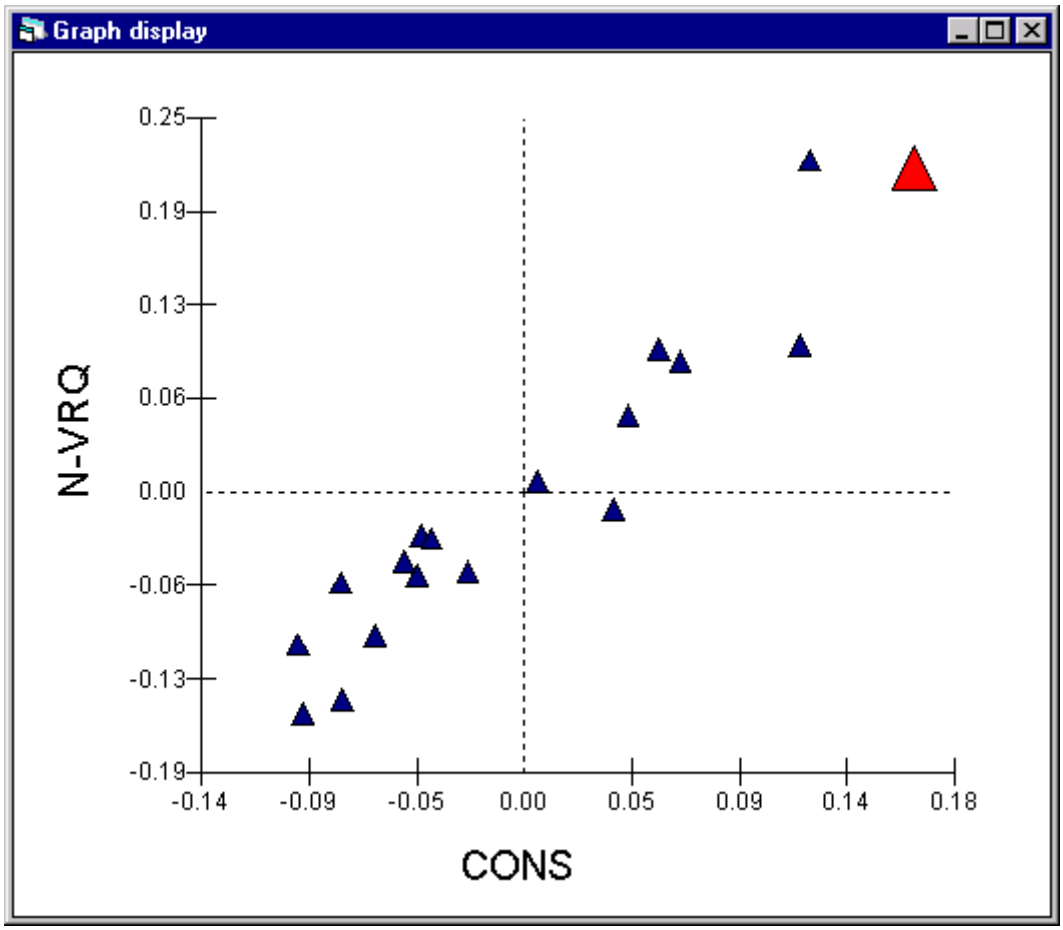
Use the button at the top of the menu to choose **Plots**, and then select the option to display **residual +/- 1.4 sd x rank**. The **Residual Plots** menu should look like this:



Choose **Apply**, and see the following graph displayed (note it is put by default into graph **D10** but you can change this option if you like):



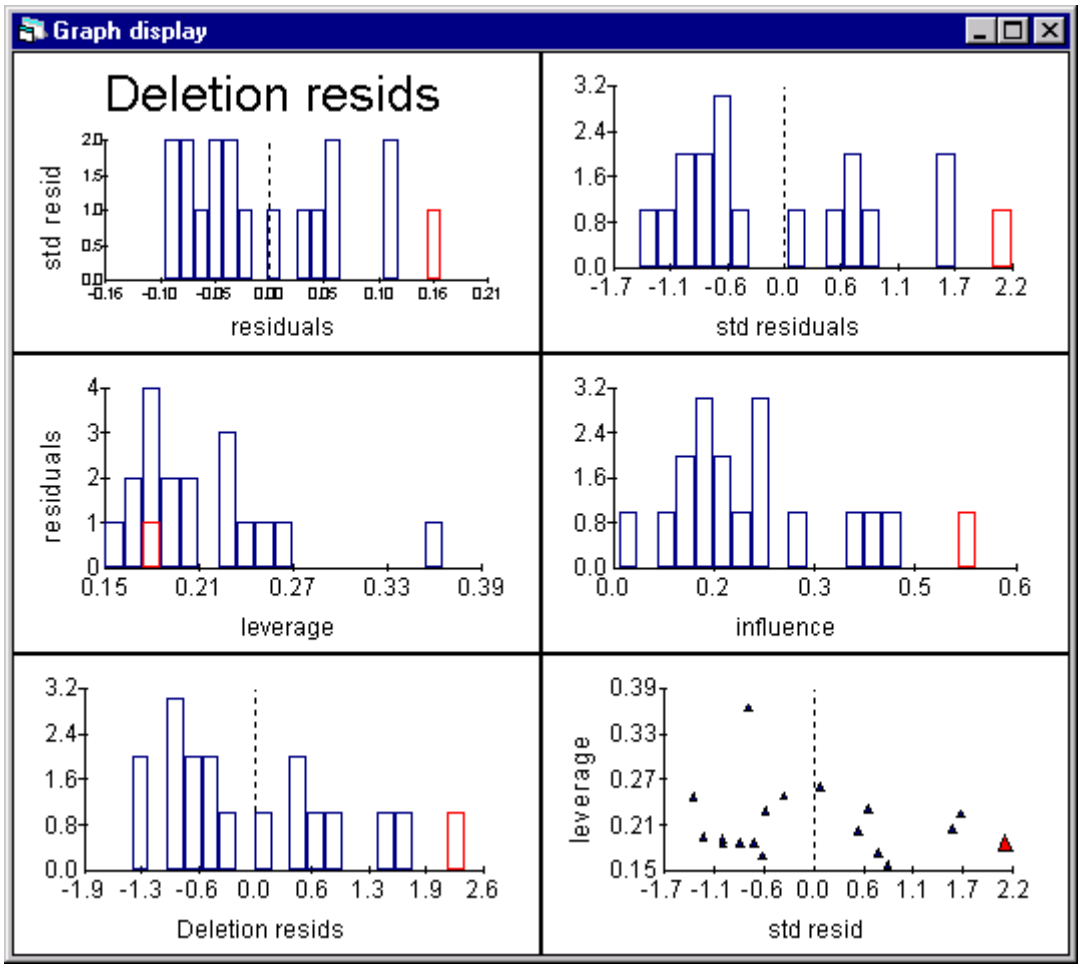
This confirms that school 17 (which we highlighted earlier) has the highest intercept residual, and the second highest slope residual. We can also examine the relationship between intercept and slope residuals. Return to the **Residual Plots** menu, and choose the **Pairwise residuals** option, then **Apply**. We get this graph as output:



We can see that there is a very strong positive relationship between the values of the intercept and slope residuals, which we can determine from the **Estimate Tables** window is 0.836. This means that the better the average performance students in a school, the more strongly positive is the relationship between outcome and intake score. School 17 again is shown in the top right hand corner of the graph.

Diagnostics plotting: Deletion residuals, influence and leverage

We can also examine a number of diagnostic measures at the school level. At the bottom of the **Residuals Plots** screen, click on the **diagnostics by variable** box, choosing CONS as the variable (this should be shown by default). Click on **Apply** and the resulting graphics window should look like this:



On this screen, we have six plots of diagnostic measures associated with the intercept at school level, the higher level in the model. School 17, which we have previously chosen to highlight, is shown in red on all six diagrams. Full explanations of the different diagnostics are given in Langford and Lewis (1998), but a brief descriptive explanation of each measure follows:

(1) The plot in the top left hand corner shows a histogram of the raw residuals at school level. As can be seen, school 17 has the highest intercept residual;

(2) The top right hand plot is a histogram of the standardised diagnostic residuals at school level (see Goldstein, 1995 for an explanation of the difference between diagnostic and comparative variances of the residuals). Again, school 17 has the highest standardised residual. The standardised residual (sometimes called the Studentised residual) is the value of the residual divided by its diagnostic standard error, and any value greater than +2 or less than –2 indicates a school which may be significantly different from the mean at the 95% confidence level;

(3) The middle left had plot is a histogram of the leverage values for each school. The leverage values are calculated using the projection, or hat matrix of the fitted values of

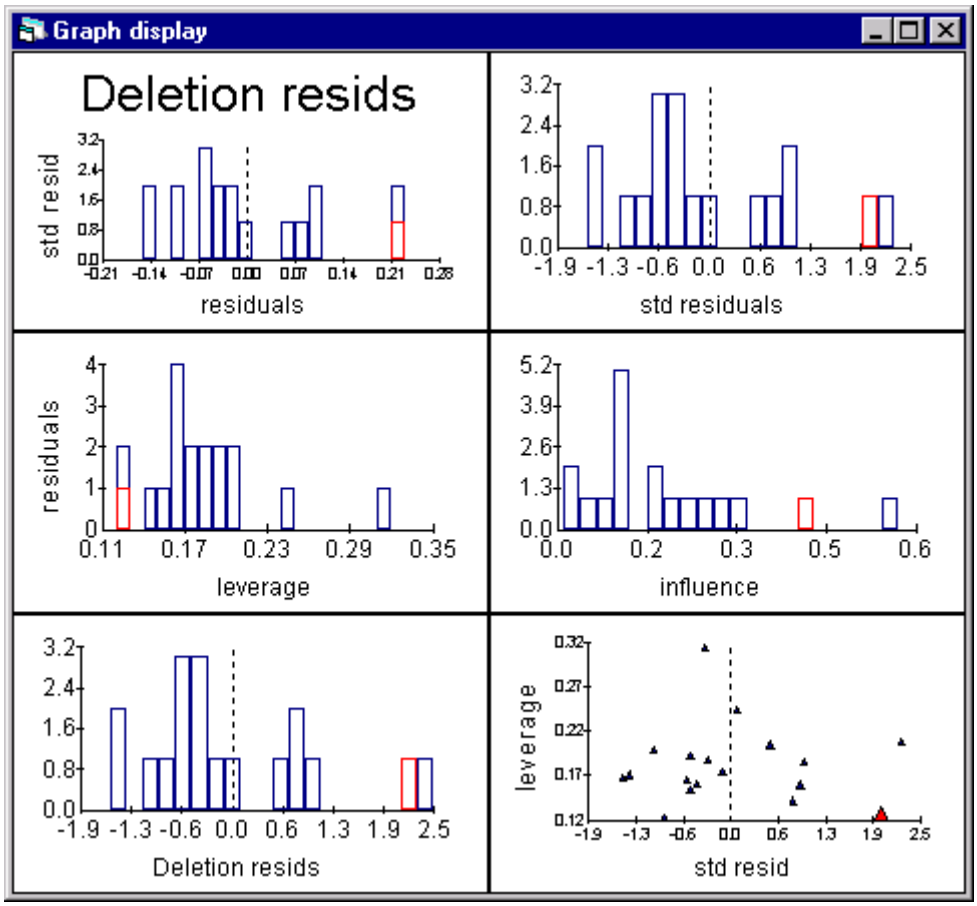
the model, and a high leverage value for a particular school indicates that any change in the intercept for that school will tend to move the regression surface appreciably towards that altered value (Venables and Ripley, 1994). An approximate cut-off point for looking at unusually high leverage values is $2p/n$, where p is the number of random variables at a particular level, and n is the number of units at that level. Here, we have 2 variables and 18 schools, so unusually high leverage values may be above 4/18, i.e. 0.22. One school, which is school 6, has a leverage value appreciably above this value, which may require further investigation. School 17 does not have a particularly high leverage value of about 0.17;

(4) The middle right hand plot shows influence values, which are a multilevel equivalent of the DFITS measure of influence (Belsey *et al*., 1980; Velleman and Welsch, 1981). The influence values combine the residuals and leverage values to measure the impact of each school on the coefficient value for, in this case, the intercept (see Langford and Lewis, 1998 for further details). School 17, having a high residual, and a roughly average leverage value comes out as having the highest influence on the intercept;

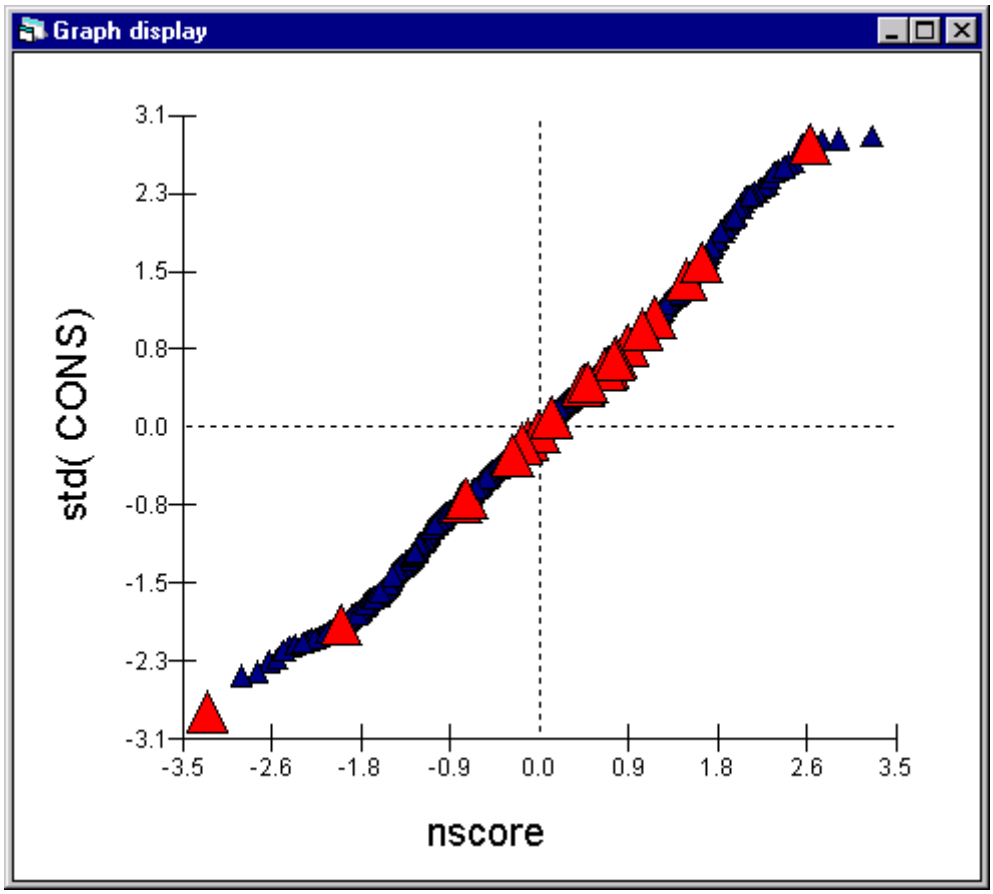
(5) The lower left hand plot is a histogram of deletion residuals. The deletion residuals show the deviation between the intercept for each particular school and the mean intercept for all schools, when the model is fitted to the data excluding that school. When the number of units at a particular level is large, these will be very similar to the standardised residuals. However, when the number of schools is small – in this case, there are only 18 schools – there may be some differences. It is the deletion residuals which are used in the calculation of influence values discussed above;

(6) The lower right hand diagram shows a plot of leverage values against standardised residuals. We can see school 17 at the far right of the graph, and school 6, with a high leverage value is towards the top left hand corner. Schools with unusual leverage values or residuals can easily be identified from the plot using the mouse.

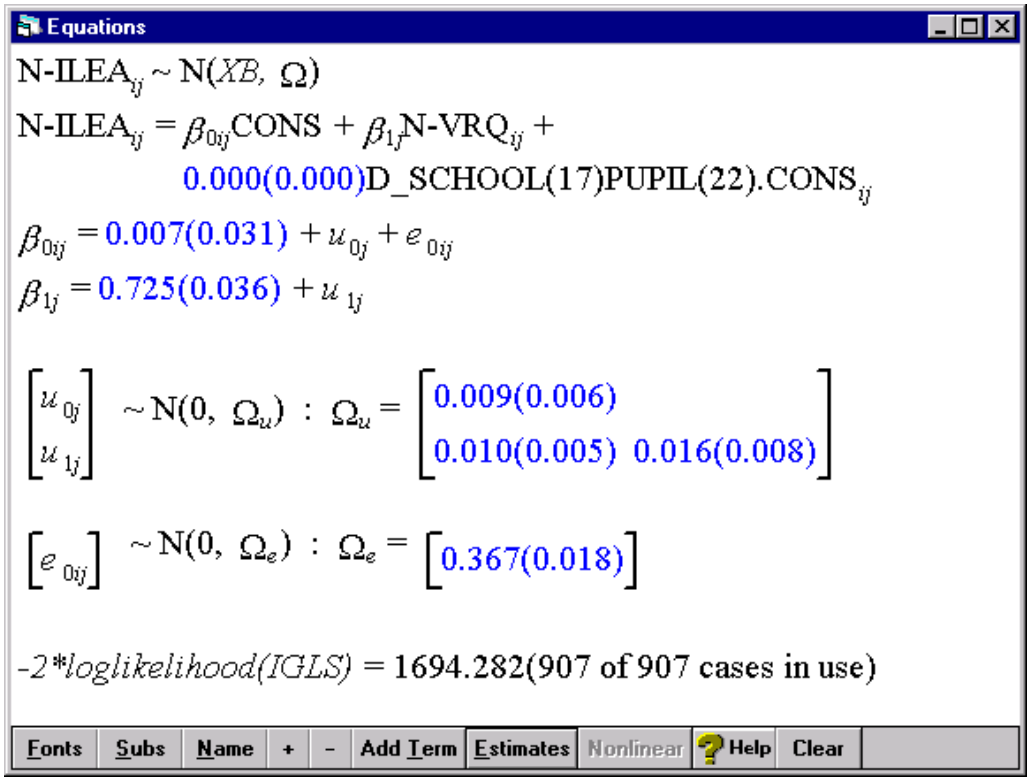
We can calculate the same measures for the slopes at school level, associated with the explanatory variable N-VRQ. If you return to the **Residual Plots** window and this time choose N-VRQ in the **diagnostics by variable** box and click **Apply**, the graph window should look like this:



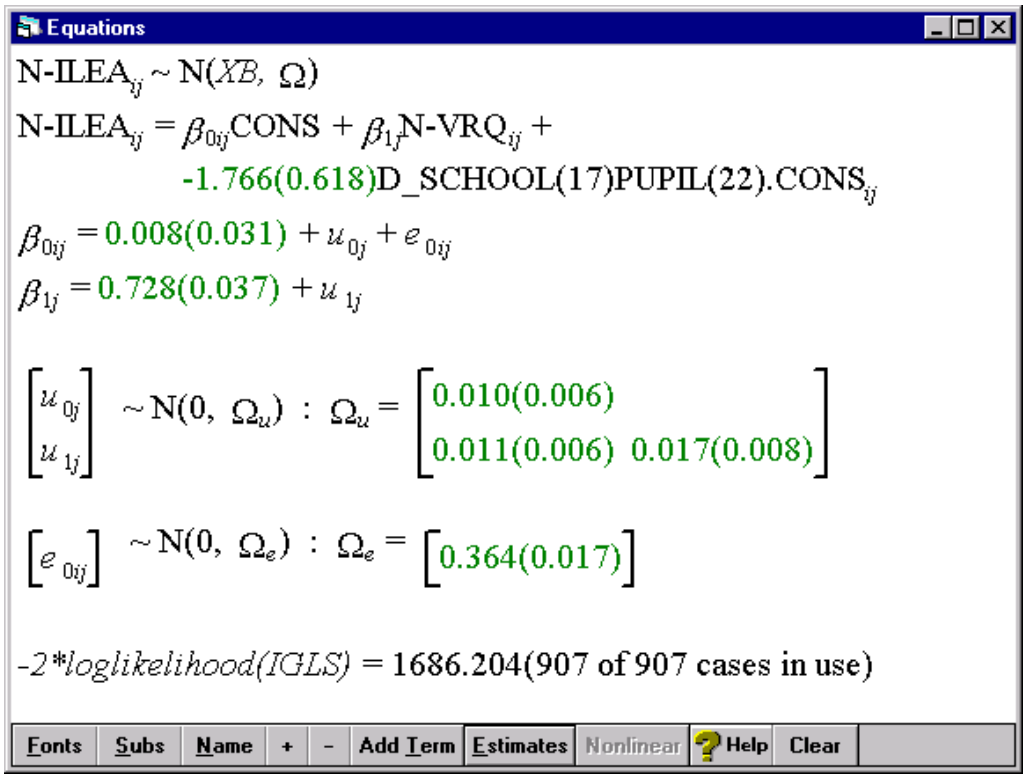
We can see from the standardised residuals plot that two schools have unusually high slope values, these being school 17 and school 12. School 12 also has the highest influence on the slope, followed by school 12, so school 12 would be a good candidate for further investigation, although for the sake of simplicity, we shall focus our attention on school 17 in this analysis. Return to the **Residuals Settings** window, and choose pupils, at level 1, as the level to calculate diagnostics, starting at c320, so we retain the columns we have used for the school level diagnostics. Return to the **Residuals Plots** menu, and choose to plot standardised residuals against Normal scores. Note that we only have one set of residuals at level 1, as only CONS is randomly varying at pupil level. The graph window should look like this:



The pupils in School 17 are shown as red triangles. As we can see, there is one pupil at the bottom left of the plot who has a particularly high negative residual, i.e. he is a low achiever in the overall high achieving school 17. We will now examine the effect on the model of omitting this low achiever from the random effects for school 17 by fitting a separate value for the low achiever in the fixed part of the model. Click the left mouse button while pointing at the red triangle for this pupil, which will pick the individual out as pupil 22 in school 17. From the **Identify Point** window, choose to highlight this pupil in **Highlight (style 2)** in the **In graphs** box (pick out the option and then click on **Apply**). Next, choose the **Absorb into dummy** option from the **In model** box, and click on **Apply**. If you return to the graph window, this particular pupil should now be shown as a light blue triangle. Now open the **Equation** window, which will have been updated to look this:

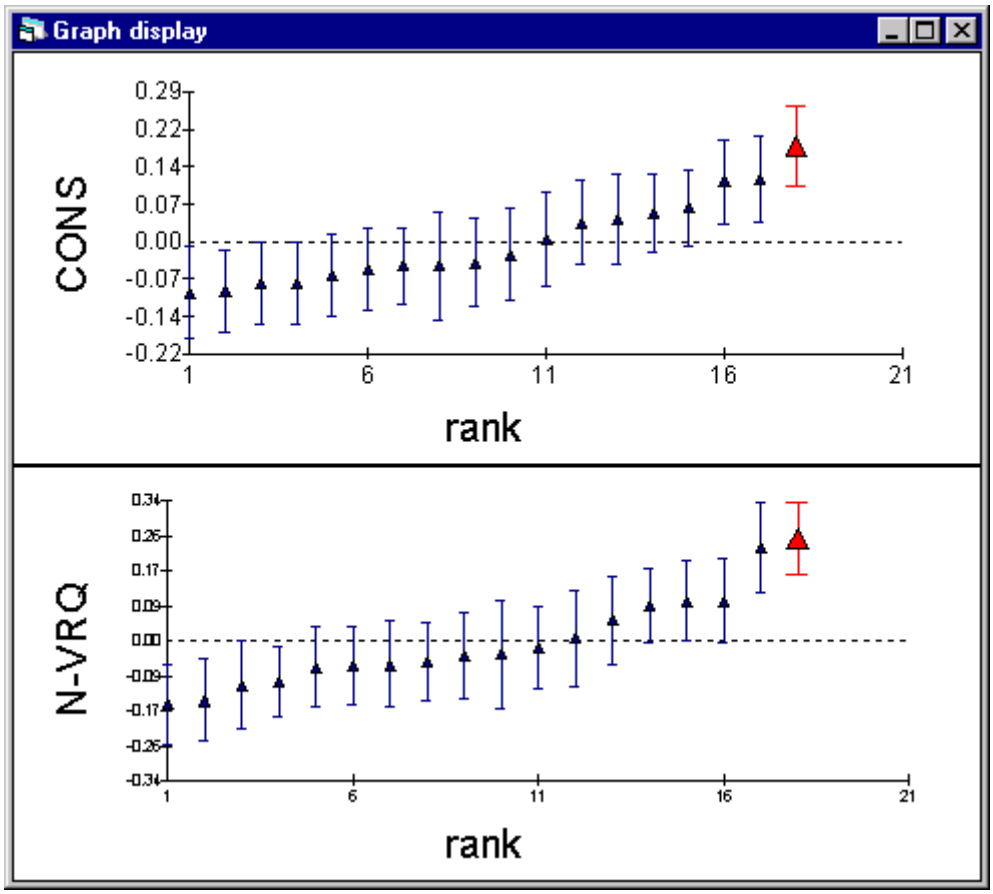


The dummy variable for school 17, pupil 22 has been added to the fixed part of model, thereby excluding this pupil from the random part of the model at pupil level. Note that the parameter estimates are now in blue, as the new model needs to be refitted to the data. Click on **More**, and when the model has converged, you should get this result:

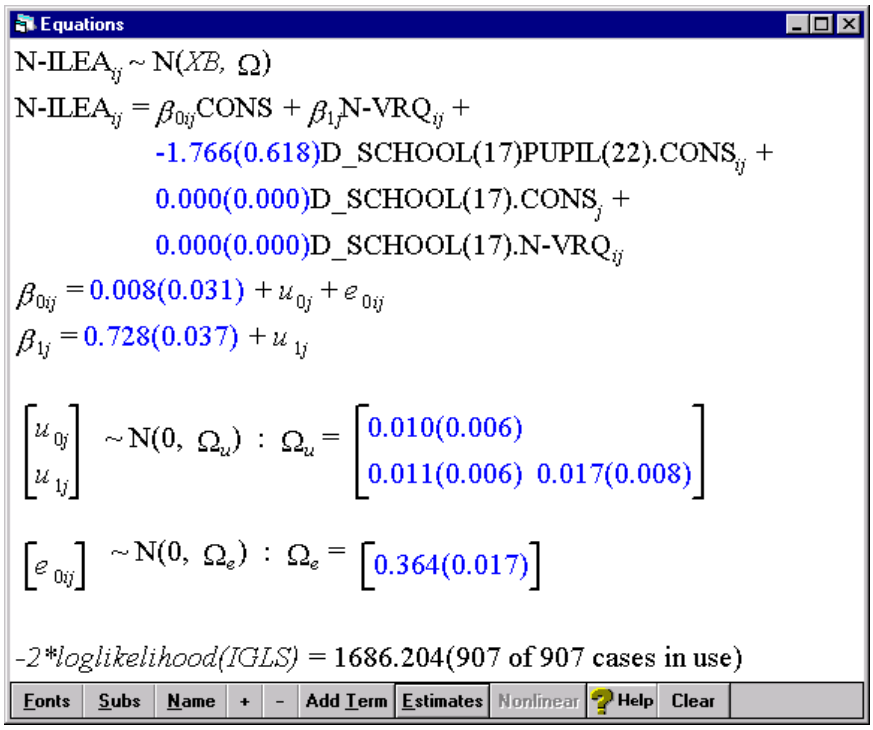


Note that the residual deviance has dropped by a little over 6 units for the inclusion of this one extra fixed parameter, a significant improvement in the model at the 0.01 significance level.

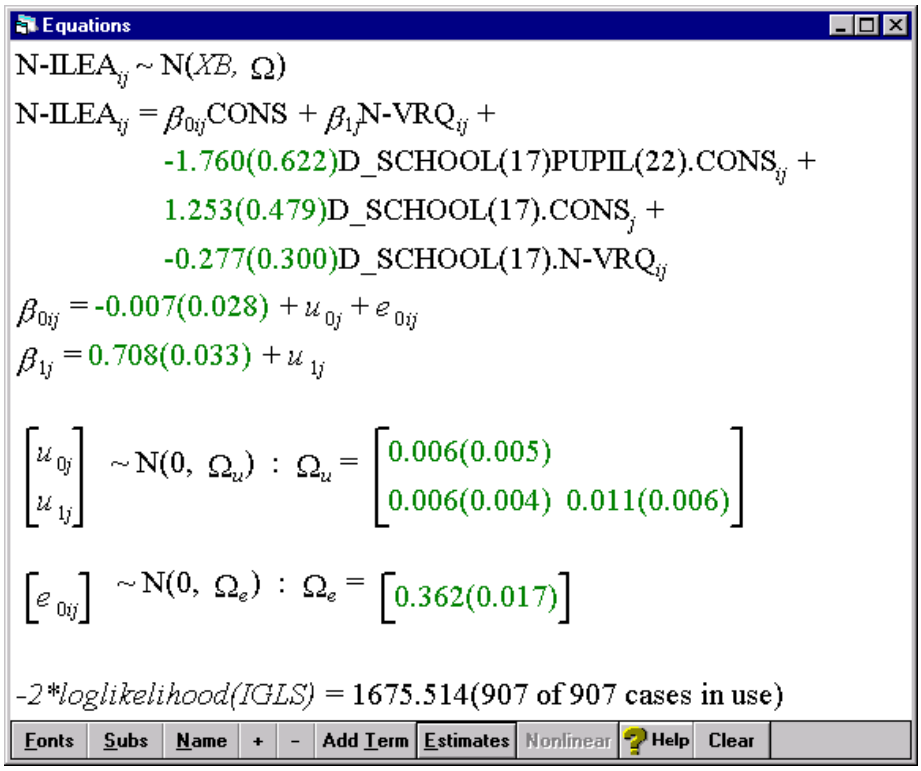
We can now re-examine the school level residuals from the new model. Return to the **Residuals Settings** window, and choose to calculate school level residuals in columns c330 onwards. Go to the **Residuals Plots** window and choose to plot **residuals +/- 1.4 sd x rank** again. The graph window should look like this:



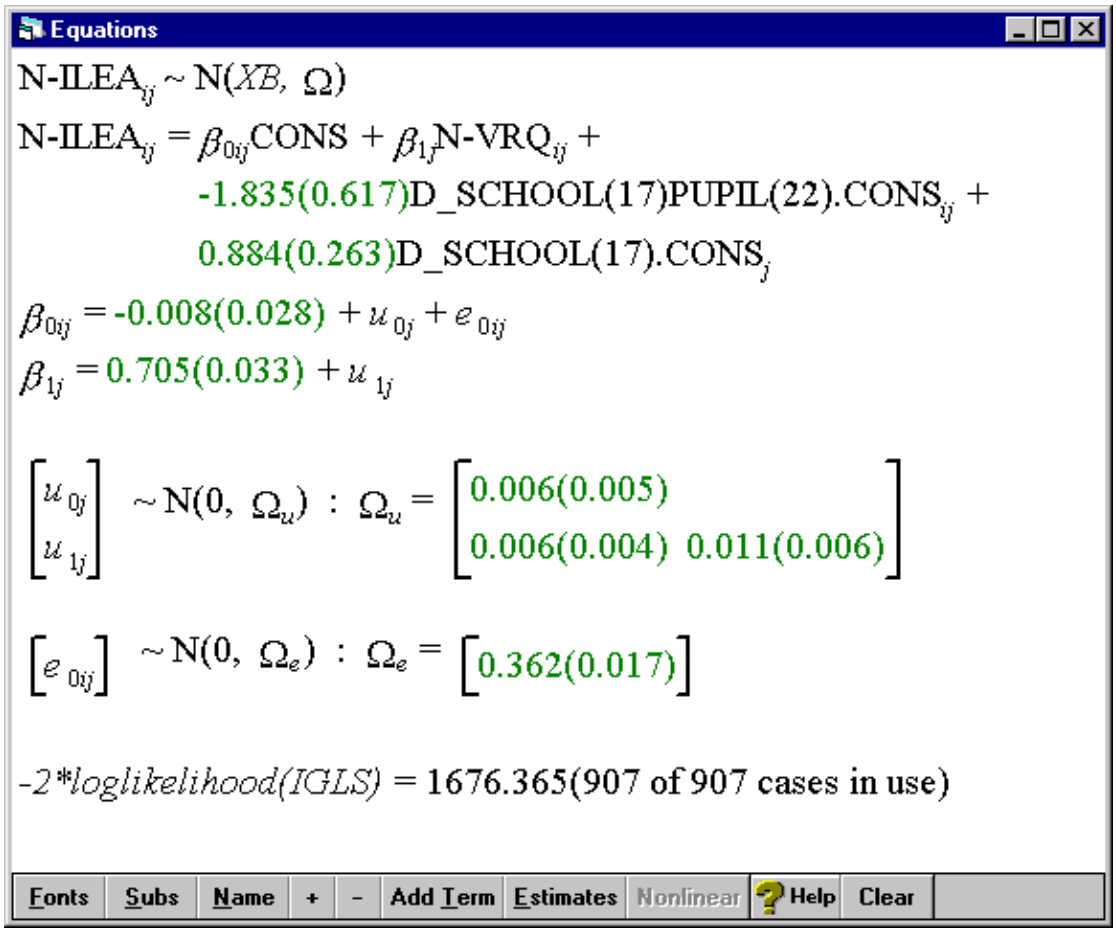
We now see that omitting the low achieving pupil *within* school 17 has made school 17 even more extreme than previously for both intercept and slope at the school level. We can choose to create dummy variables for school 17, to fit a separate intercept and slope to the other schools by selecting it on the graph with the mouse, clicking on the left hand button, and using the **Identify point** window, as before. Choose the **Absorb into dummy** variable from the **In model** box again, and this time when you click on **Apply**, another window will open, asking whether you want to fit a dummy variable for CONS and/or N-VRQ. Make sure both variables have 'tick-marks', and click on **Done**. Return to the **Equation** window, which should now look like this:



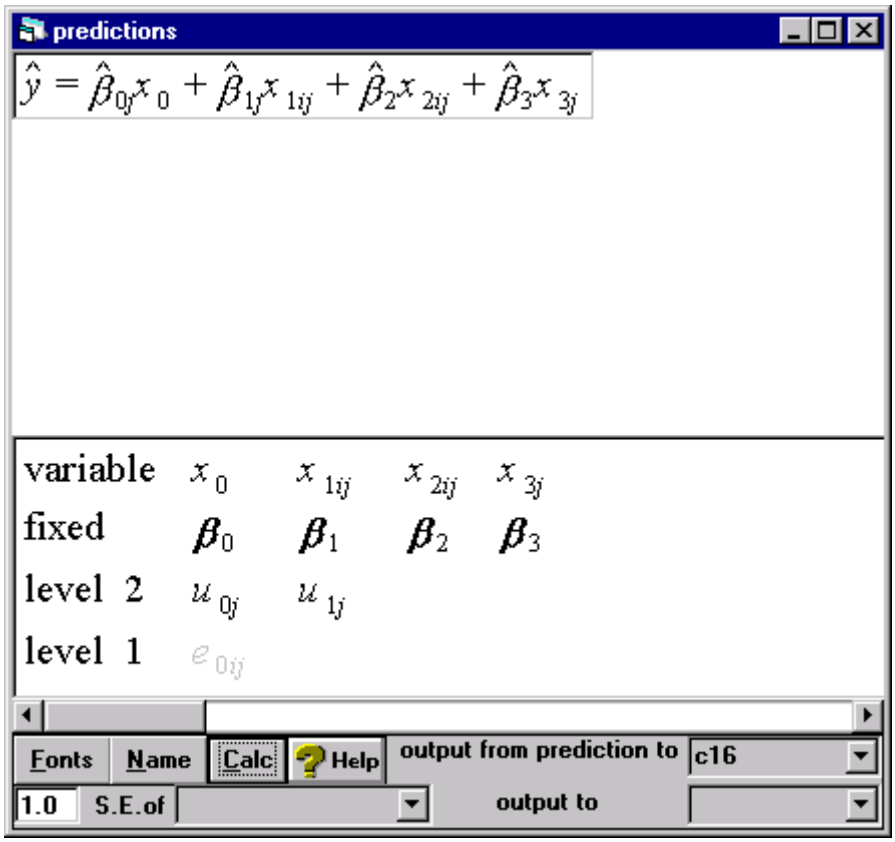
We can see that two extra variables have been added into the model, an intercept term for school 17, and a slope term. Click on **More** and wait for the model to converge. The result should be:



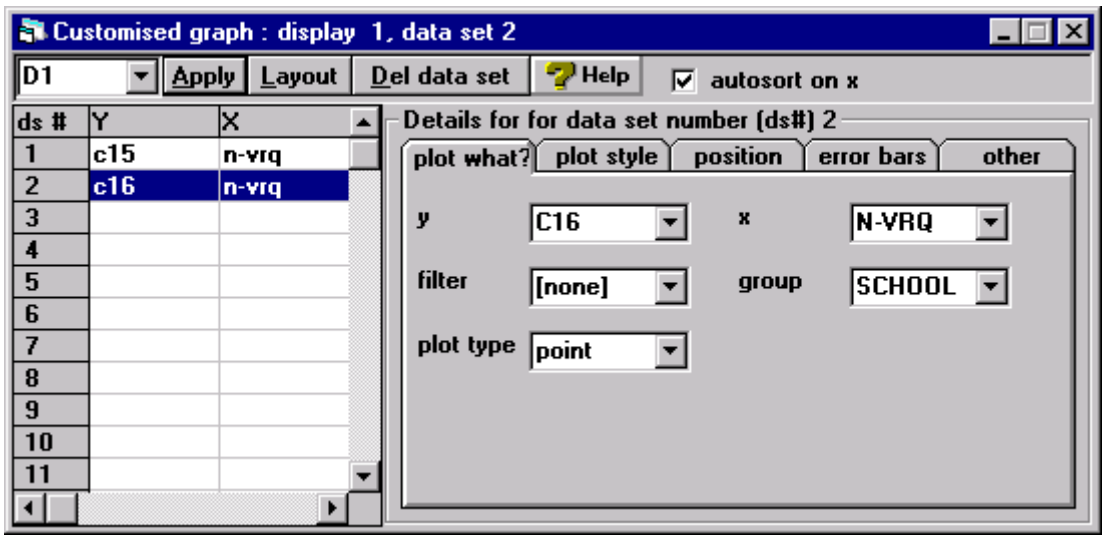
Note that the random parameter estimates at school level for the variances of slopes and intercepts have both dropped noticeably, now that the most extreme school has been excluded from the random part of the model. The overall decrease in residual deviance is about 10.7, with the loss of two degrees of freedom for the extra variables included. This is highly significant, but if we look at the individual parameter estimates, we can see that the intercept value for school 17 is highly significant, whilst the slope value is not. We can examine the effect of excluding the slope variable for school 17 by clicking on the variable in the **Equations** window, and choosing to **delete term** from the model. Refit the model using **More**, and the converged model should look like this:



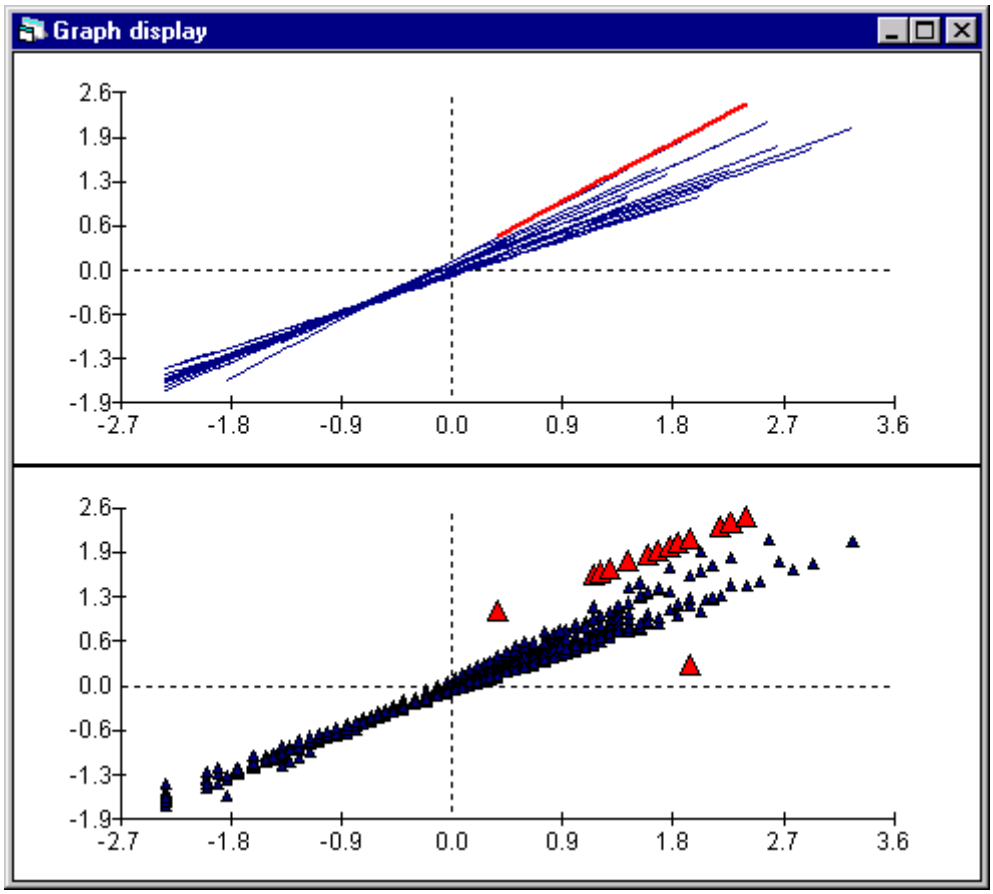
The residual deviance has only changed by a small amount, so we conclude that this is a more parsimonious model to fit to the data, with a separate intercept for school 17, and a separate fixed effect for pupil 22 in school 17. We can examine the predictions for the school by using the **Predictions** window as before, which you should set up to look like this (save the predictions into c16) before clicking on **Calc**:



Now go to the **Customised graph** window, and choose graph **D1** as before. Leave data set 1 as it was, and choose to edit data set 2 to show the new predictions from the current model. The window should look this before you click on **Apply**:



Note that we are choosing to plot points, rather than lines. The resulting graph should look as below, showing at the top our initial predictions from the model as a line graph, with school 17 highlighted, and at the bottom, our new predictions, with pupils in school 17 highlighted as red triangles. Note the pupil which does not fit on the line, which is pupil 22, the one we have chosen to exclude from the random part of the model. Note also that there is another low achieving pupil that we could choose to exclude from the model (see Langford and Lewis, 1998 for more details on this).



We can then explore the regression diagnostics, choosing to examine particular schools or pupils as before in more detail.

A general approach to data exploration

We may be interested in observations which are outlying with respect to any of the random parameters in a model, and hence the first issue is where to start data exploration in a multilevel model. Rather than looking at individual data points, we have found it most useful to begin at the level of highest aggregation, which will often be simply the highest level in the model. There are two reasons for this. Researchers are often most interested in the highest level of aggregation, and will naturally concentrate their initial efforts here. However, if discrepancies can be found in higher level

structures, these are more likely to be indicative of serious problems than a few outlying points in lower level units. After the highest level has been analysed, lower levels should be examined in turn, with analysis and initial treatment of outliers at the lowest level of the model. The highest level should then be re-examined after a revised model has been fitted to the data. The objective is to determine whether an outlying unit at a higher level is entirely outlying, or outlying due to the effects of one or two aberrant lower level units it contains. Similarly, examination of lower level units may show that one or two lower level units are aberrant *within* a particular higher level unit which does not appear unusual, and that the higher level unit would be aberrant without these lower level units. Hence, care must be taken with the analysis not simply to focus on interesting higher level units, but to explore fully lower level units as well.

What you will have learnt from this chapter

- The use of diagnostic procedures for exploring multilevel models
- The importance of studying data carefully to check model assumptions
- How to deal with 'discrepant' measurements

PART 3. SIMULATION BASED METHODS

Chapter 13: An Introduction to Simulation

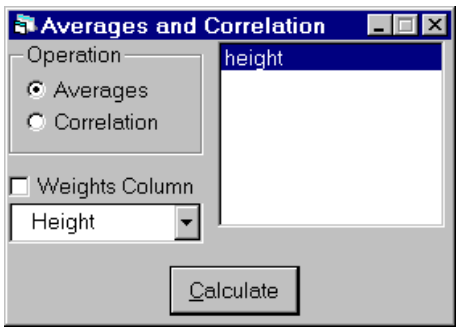
In part 1 of this manual the concepts of multilevel modelling were introduced and it was shown how to fit some simple Normal models in *MLwiN*. In part 2 we went on to look at other types of models and how these could also be fitted in *MLwiN*. In both these sections the emphasis was on how to fit models, how to choose between models and the interpretation of the models. In all examples default estimation methods were used and little mention was given to alternative approaches to fitting multilevel models.

In this part of the manual we will be looking at two other groups of methods that can be used, namely MCMC methods and bootstrap methods. Both these groups of methods are based on simulation techniques and the estimates they produce are dependent on random numbers. Consequently using a different set of random numbers, or a longer simulation run can produce (slightly) different estimates. For this reason, and because these methods are fairly new we include more theory in this section of the manual. This additional theory is not included to scare off the new user but instead to enhance their knowledge of the techniques they are using. Although it is not necessary to understand all the theory to use the simulation methods in *MLwiN*, care must be taken to ensure that the simulation methods have been run for long enough to give accurate answers. This will explained further in the later chapters but first, to start this section we will give a brief introduction to simulation methods in general.

Normally distributed data

Probably the most commonly studied data in the field of statistics are where variables are continuous, and the most common distributional assumption for continuous data is that they are Normally distributed. The example of a continuous dataset that we will consider here is the heights of adult males. If you were asked to estimate the average height of adult males in Britain how would you provide a good estimate? One approach would be to take a simple random sample, that is travel around the country measuring a sample of the population. Then from this sample we could calculate the mean and use this as an estimate.

The worksheet height.ws has 1 column of data, c1, named **height** that contains the heights of 100 adult males measured in centimetres. Having loaded up this worksheet using the **Open Worksheet** option we can calculate the average height of the data via the **Averages and Correlations** window that can be found under the **Basic Statistics** menu.



Select **height** from the column list on the right of the window and click on **Calculate**

The Output window will appear and the following results are given for our sample:

->AVERage 1 "height"

	N	Missing	Mean	s.d.
Height	100	0	175.35	10.002

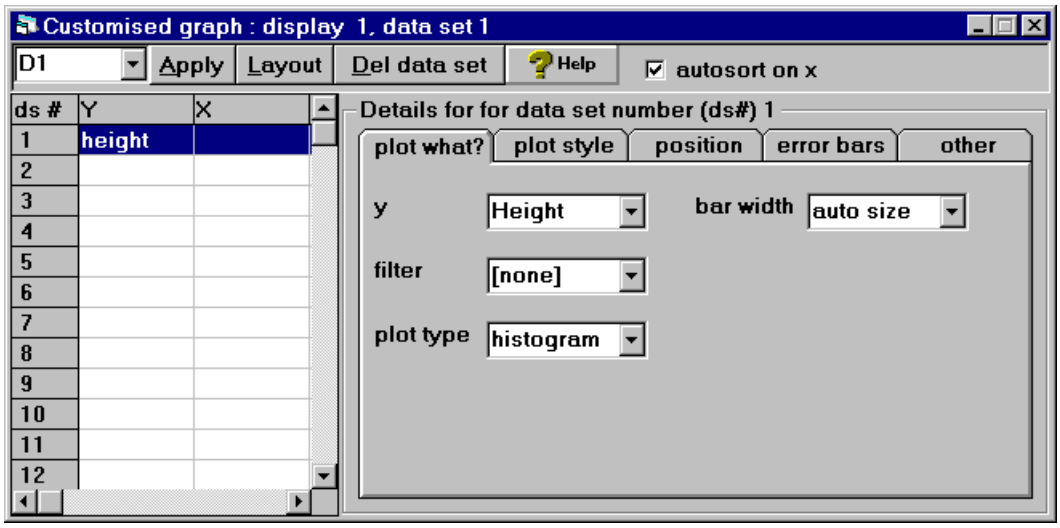
Obviously the larger the sample of people that are measured, the more accurate the mean estimate will be, and consequently the better estimate the sample mean will be for the population mean. We can also plot the 100 heights as a histogram to give a graphical description of this dataset.

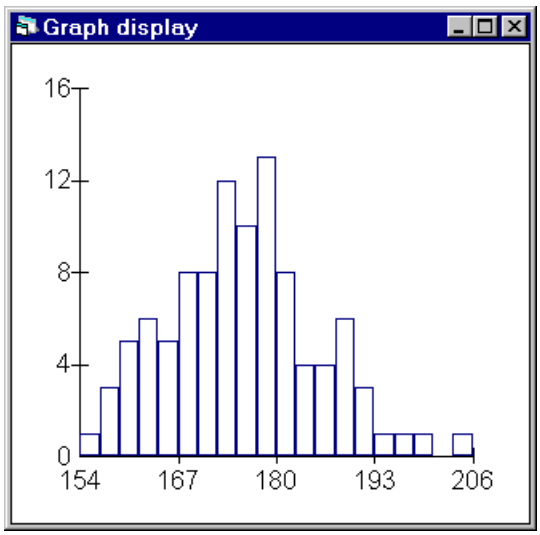
Select the **Customised graph** window from the **Graphs** menu.

Select **Height** from the **y** list.

Select **histogram** from the **plot type** list.

Click on **Apply**.





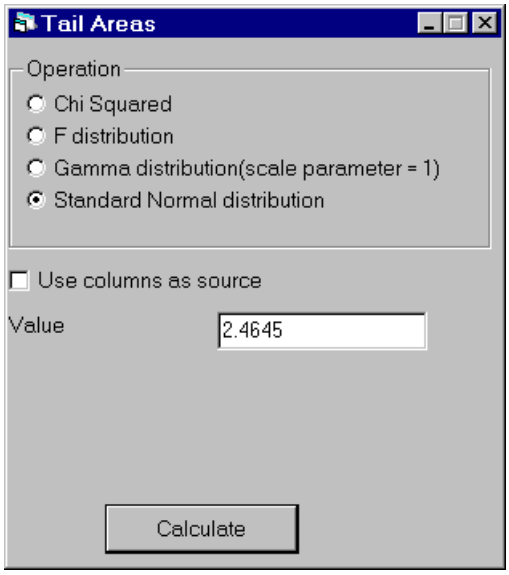
The histogram will now appear as shown above. Note that the shading pattern for the bars and their colour can be altered in the options under the **plot style** tab.

Another question that could be asked is 'What percentage of British adult males are over 2 metres tall?' We can consider two approaches to answering this question, a *non-parametric* approach or a *parametric* approach. The non-parametric approach would be to calculate directly from the list of 100 heights the percentage of heights that are above 2 metres. In this case the percentage is 1% as only 1 height is greater than 2 metres. The parametric approach would involve making an assumption about the distribution of the data. Typically we would assume that the data are Normally distributed, which from the histogram of the 100 heights looks feasible.

So we would then need to find the probability of getting the value 200 from a Normal distribution with mean 175.35 and variance 10.002. *MLwiN* allows the user to find probabilities for a standard Normal distribution so we will need to Z transform our value via the calculate window to give (200-175.35)/10.002 = 2.4645.

Then we can use the **Tail Areas** window from the **Basic Statistics** menu as follows:

Select Standard Normal distribution from the **Operation** list Type 2.4645 for the **Value** box Click on **Calculate**



This gives the value

->NPRObability 2.4645 0.0068602

and so the parametric approach estimates that 0.68% of the population are over 2 metres tall. We will be considering parametric and non-parametric approaches again when we consider the bootstrap methods later.

Distributions of the mean and variance

We have not as yet used any simulation-based methods. We will now consider performing inference on the population mean and variance that our sample of 100 people have been taken from in more detail and attempt to calculate the distributions of these quantities. Here we will consider 3 methods

Normal Distribution Sampling Theory

In the case of a sample of Normally distributed observations of size N distributions of the population mean, μ and variance σ^2 can be calculated from sampling theory.

The sample mean, $\bar{x}$ is Normally distributed with mean μ and variance σ^2/N. Consequently a 95% central confidence interval for μ is $\bar{x} \pm 1.96\ \sigma/\sqrt{N}$. In our sample of people's heights a 95% central confidence interval for μ is 175.35 ± 1.96 * 10.002/√100 = (173.39, 177.31).

The population variance, σ^2 is related to the sample variance, s^2 by the Chi-squared distribution as follows $(N\text{-}1)\ s^2 / \sigma^2 \sim \chi^2_{N-1}$. Consequently a 95% central confidence interval for σ^2 is $((N-1)s^2/\chi^2_{(N-1),0.025}, (N-1)s^2/\chi^2_{(N-1),0.975})$. In our sample a 95% central confidence interval for σ^2 is (77.12,135.00) which is not symmetric.

Bootstrapping

For the mean and variance a single sample gives us one estimate for each parameter. If we could get a *sample* of mean estimates and a *sample* of variance estimates then we could use these samples to construct interval estimates for the underlying parameters. This idea of generating a large number of samples to create interval estimates is the motivation behind most simulation methods.

Bootstrapping works by constructing lots of datasets similar to our actual dataset (using the actual data set as an estimate of the population distribution) and then using these datasets to summarise the parameters of interest. The way the datasets are constructed depends on which type of bootstrapping is used.

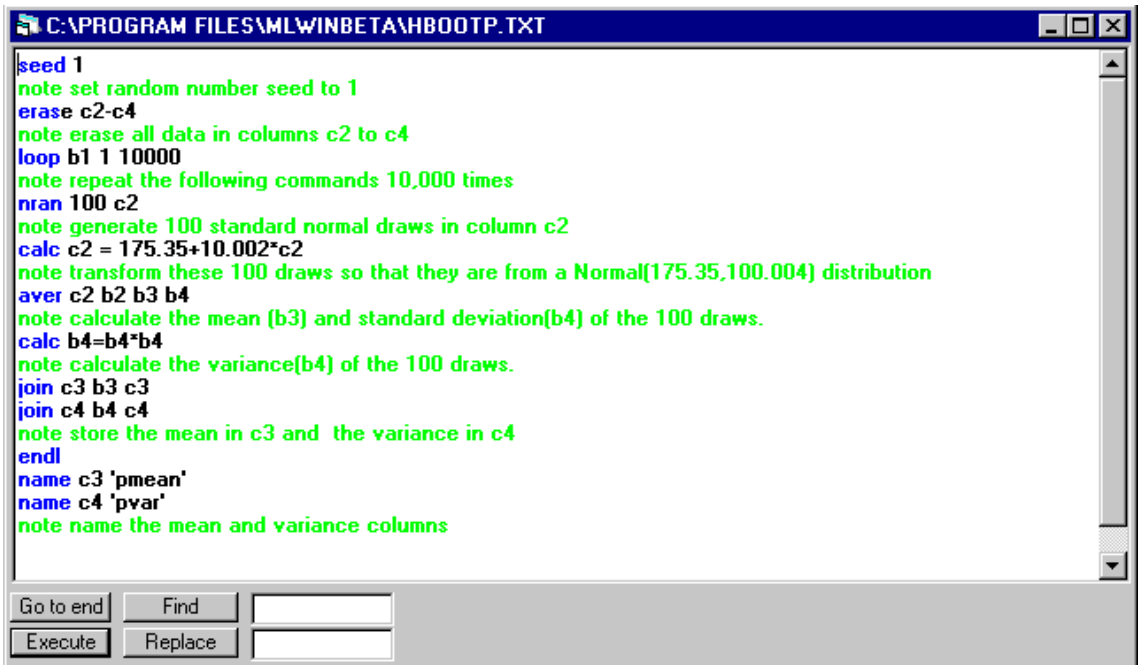
The parametric bootstrap

Parametric bootstrapping uses assumptions about the distribution of the data to construct the bootstrap datasets. Consider our sample of 100 heights that has a mean of 175.35 and a standard deviation of 10.002. To create parametric bootstrap datasets from this dataset we simply draw lots of samples of size 100 from a Normal(175.35, $(10.002)^2$) distribution. Then for each sample we calculate the parameter we are interested in. To illustrate this bootstrapping procedure in *MLwiN* we will introduce the *MLwiN* macro language that can be used to run a series of commands in *MLwiN*.

Select the option **Open Macro** from the **File** menu.

From the list of files select **hbootp.txt** and press **Open**.

The window shown below should now appear.



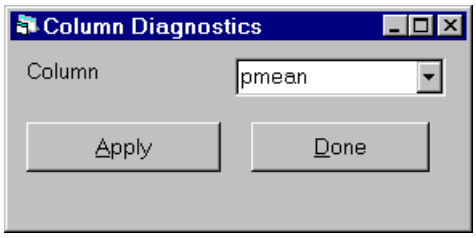
Virtually all menu operations in the *MLwiN* have an equivalent command that can be typed into the **command interface** window or executed via a macro. These commands were used in the *MLn* package that was a forerunner for *MLwiN*. For more information on any of the commands if you look in the online help under Command, you will find listings for each command under suitable headings.

The macro above is designed to generate 10,000 samples of size 100 from a Normal distribution with mean 175.35 and standard deviation 10.002. Then for each sample the mean is stored in column c3 named ‘pmean’ and the variance in column c4 named ‘pvar’.

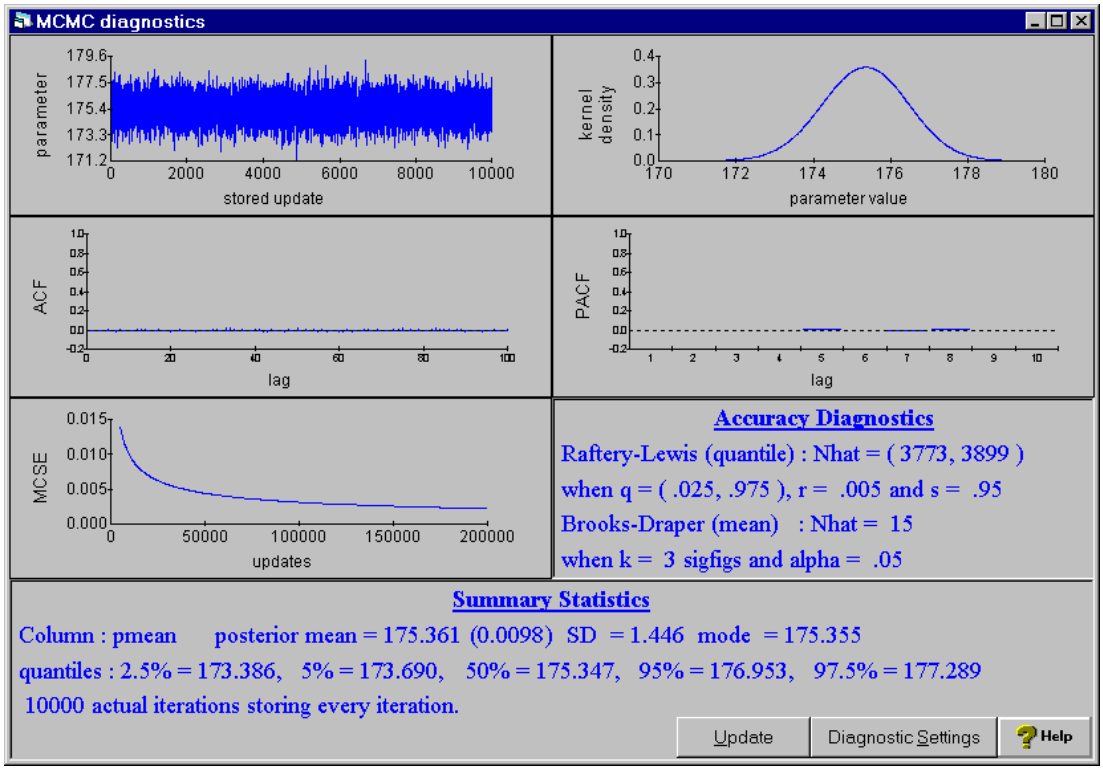
You will notice that all of the *MLwiN* commands are highlighted in blue, if you want to know more about any particular command you can find details in the help system. Highlighting a command in the macro window and pressing F1 will bring up the help entry for that command. You will also see that the **note** command lines are highlighted in green. The **note** command is used to add comments to a macro and all lines starting with **note** are ignored when the macro is run.

Now click on **Execute** to run the macro. After a short time the mouse pointer will change back from an hourglass symbol to a pointer to signify that the macro has finished. We would now like to look at the chains of mean and variance values in more detail.

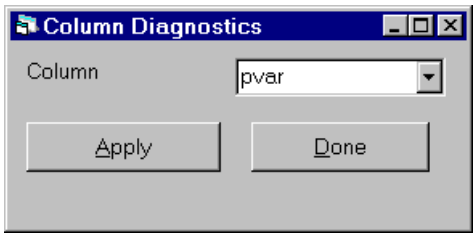
Select the **Column Diagnostics** window from the **Basic Statistics** menu and select from the column list the column labelled **pmean** as can be seen in the window below.



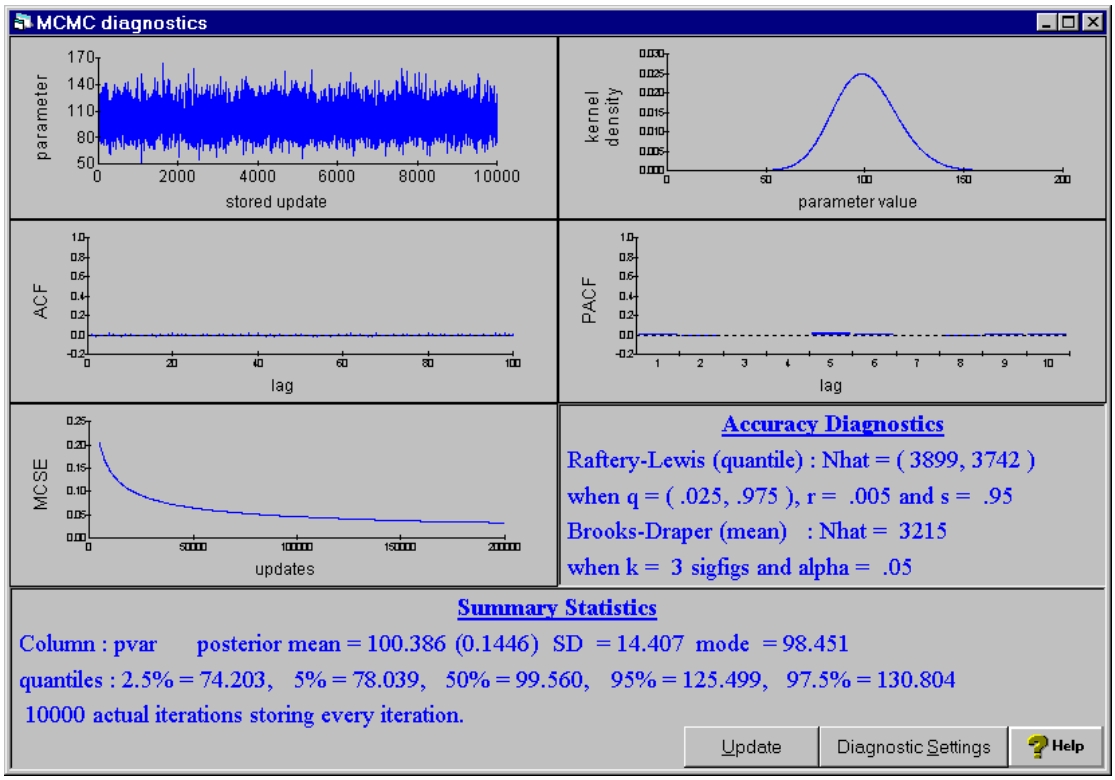
Now press the Apply button and after a short wait the MCMC diagnostics window will appear as below. This diagnostics window is generally used for MCMC chains as will be discussed in later chapters so a lot of the information on the window is irrelevant at this point.



We will concentrate on the top two graphs and the summary statistics box. The graph on the left is a trace plot of the means of the 10,000 samples in the order that they were generated. These 10,000 values have been used to construct the kernel density graph on the right. A kernel density plot (Silverman 1986) is like a smoothed version of a histogram where instead of allocating each value to an appropriate histogram bin, its contribution to the graph is instead allocated smoothly via a kernel function. As we can see the mean appears to be Normally distributed which is what we expected. The bottom box contains various summary statistics for the mean parameter, including quantiles that can be used to construct an interval estimate. Here a 95% central interval is (173.39,177.29) which compares favourably as it should with the theoretical interval of (173.39,177.31). We will also look at the population variance. Here we will need to select the variable **pvar** from the **Column Diagnostics** window as shown below



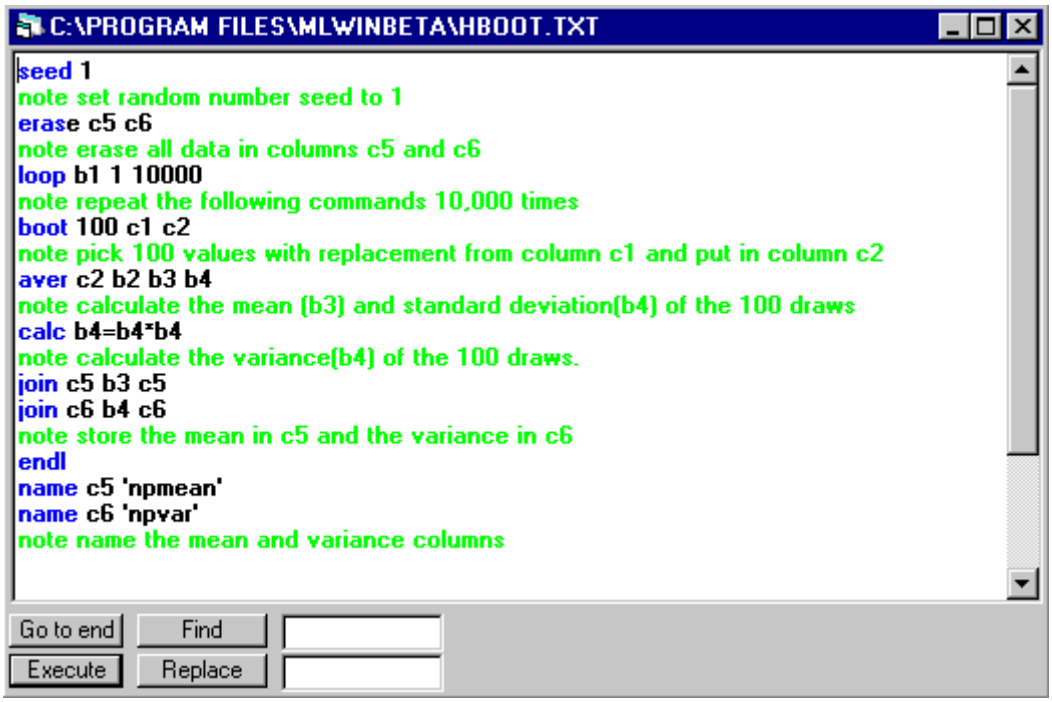
Clicking on the Apply button will after a slight delay bring up the MCMC diagnostics window for the variance parameter.



Here we see that the kernel density plot does not now look quite like a Normal distribution and has a slightly longer right hand tail. This was also to be expected based on the theoretical distribution of the variance. Now if we look at the confidence interval constructed by the quantiles we get (74.203,130.804) which is similar to (77.120,135.003) but not as close as when we considered the mean parameter. The method of taking the quantiles from the chains of the distribution is known as the percentile method in the bootstrapping literature (see Efron and Tibshirani (1993) for more details). This technique is known to be biased for parameters with skewed distributions and with small samples, and other methods for example the BCA method can be used instead, but they will not be discussed here.

Non-Parametric Bootstrap

Another type of bootstrapping that we can use on this problem is non-parametric bootstrapping. Here we do not assume a distribution for the data but instead generate a large number of datasets by sampling (with replacement) from the original sample. In our example we will again generate samples of size 100 and will write a macro to perform the sampling. This macro has already been written and can be accessed via the **Open Macro** option under the **File** menu. This time select the file **hboot.txt** and select **Open** and the following window will appear.



This macro is rather similar to the macro for parametric bootstrapping except for how each sample is generated. In the earlier macro we used the NRAN and CALC commands to construct each bootstrap dataset from the correct Normal distribution. Here we use the BOOT command:

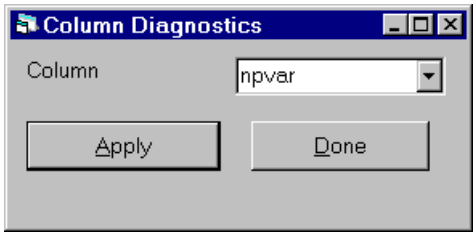
BOOT 100 c1 c2

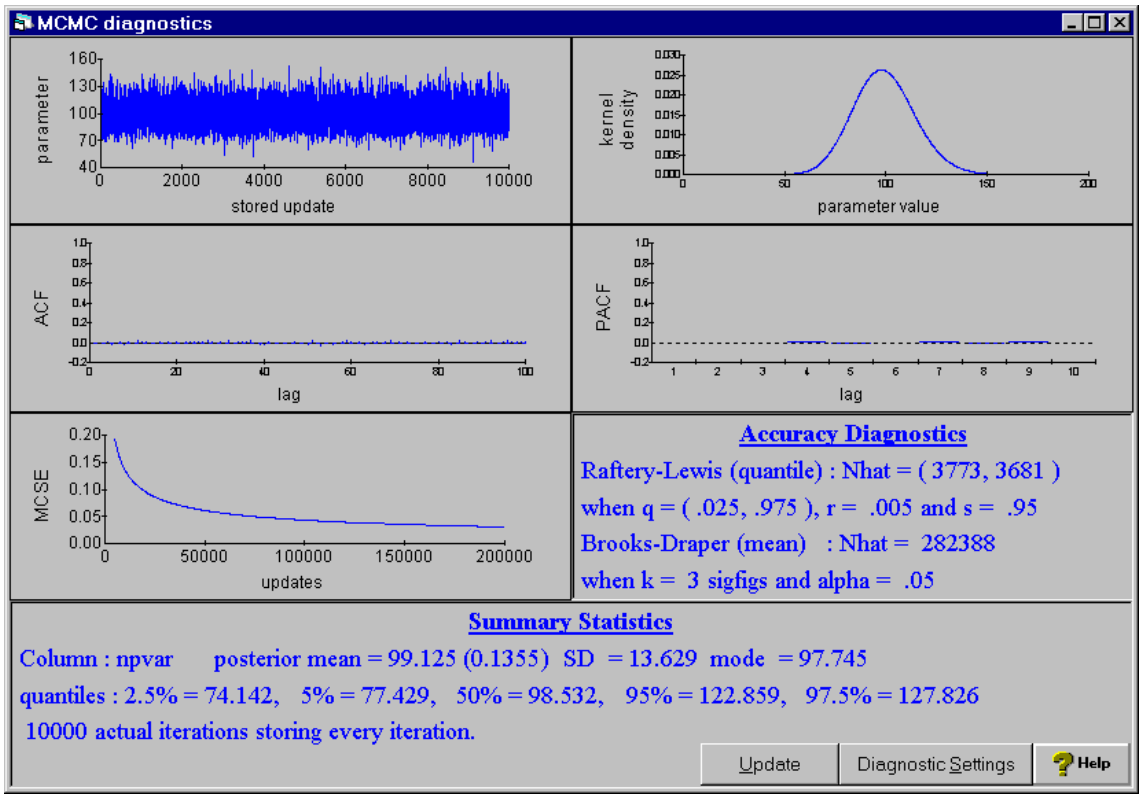
This command constructs a sample of size 100 in column c2 by sampling with replacement from the values in column c1.

Click on **Execute** and the macro will run putting the 10,000 means into column c5 named ‘npmean’ and the 10,000 variances into column c6 named ‘npvar’.

Again we can look at the summaries of both of these parameters using the column diagnostics window. The mean is not shown here but gives a central interval

(173.37,177.32) which is approximately equal to the Normal theory interval of (173.39,177.31) as would be expected due to the central limit theorem. To look at the variance parameter, we select the **Column Diagnostics** window from the **Basic Statistics** menu. Having selected **npvar** from the list as shown below, clicking on Apply will bring up the MCMC diagnostics window as shown below.





Here we again see that the kernel density plot shows a slight skew to the right. The 95% central confidence interval from the percentile method is (74.142,127.826). This is only slightly different from the interval from the parametric method which shows that the Normality assumption is acceptable in this case.

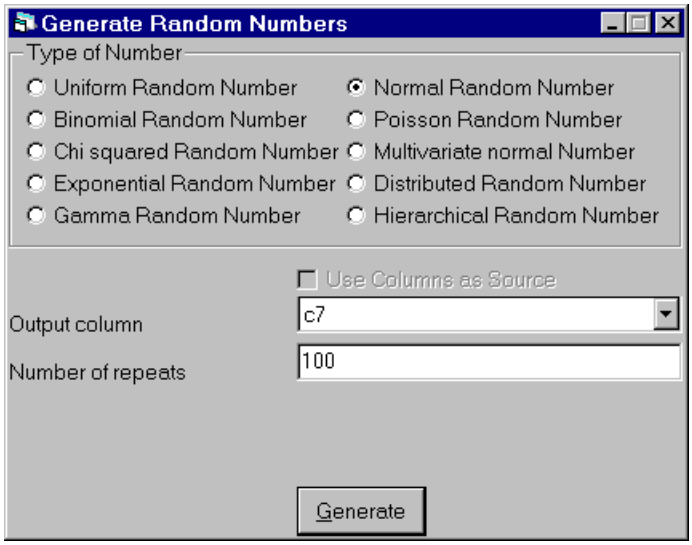
Having seen this simple example and seen how easy it is to calculate theoretical intervals for the mean and variance of a Normal distribution you may be asking yourself what is the point of simulation methods such as bootstrapping. As we have seen in earlier chapter multilevel models are far more complex than this example. In most multilevel models no simple formula exists for the distributions of the unknown

parameters and iterative routines are required to get estimates. In models like these simulation techniques come into their own as even though a simple formula does not exist, it is often easy to generate by MCMC and bootstrapping methods simulated values from the distributions of the parameters of interest and hence calculate point and interval estimates.

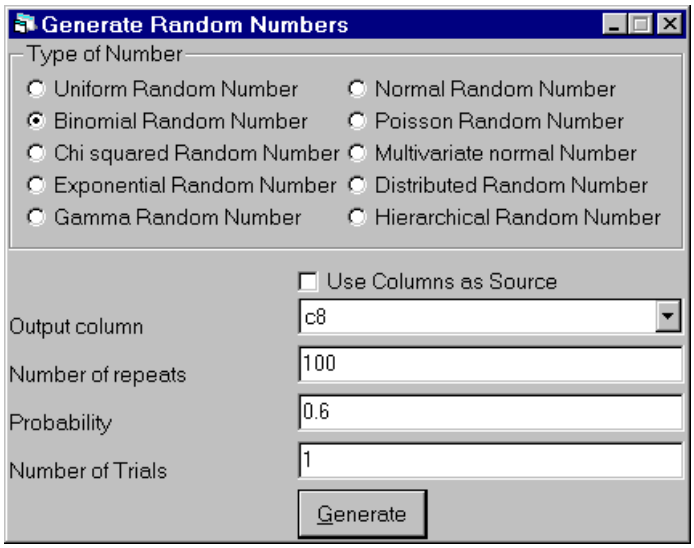
Generating random numbers in *MLwiN*

The above height dataset was actually generated by simulating from a known Normal distribution and rounding the heights to the nearest cm. *MLwiN* allows you to generate random numbers from several common distributions. To demonstrate this we will consider another example. Here we will still consider a dataset of people's heights but this time we will consider a mixed population of men and women. We will assume that the men are Normally distributed with a mean of 175cm and a standard deviation of 10cm while the women have a mean of 160cm and a standard deviation of 8cm. We will also assume that men make up 40% of the population while women make up the other 60%. We will now show how to use the **Generate Random Numbers** screen in *MLwiN* to create a sample of 100 people from this population.

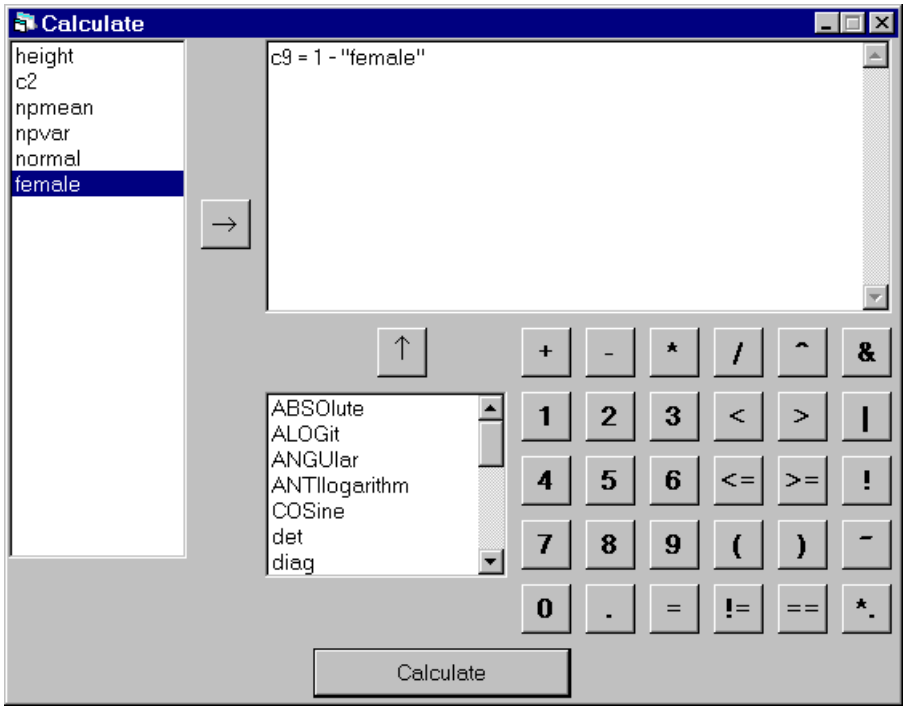
We will first generate 100 standard Normal random numbers from which we will construct our sample. Select the Generate Random Numbers option from the Basic statistics menu. Then select the type of number option **Normal Random Number**, set Output column to c7 and number of repeats to 100 as shown below.



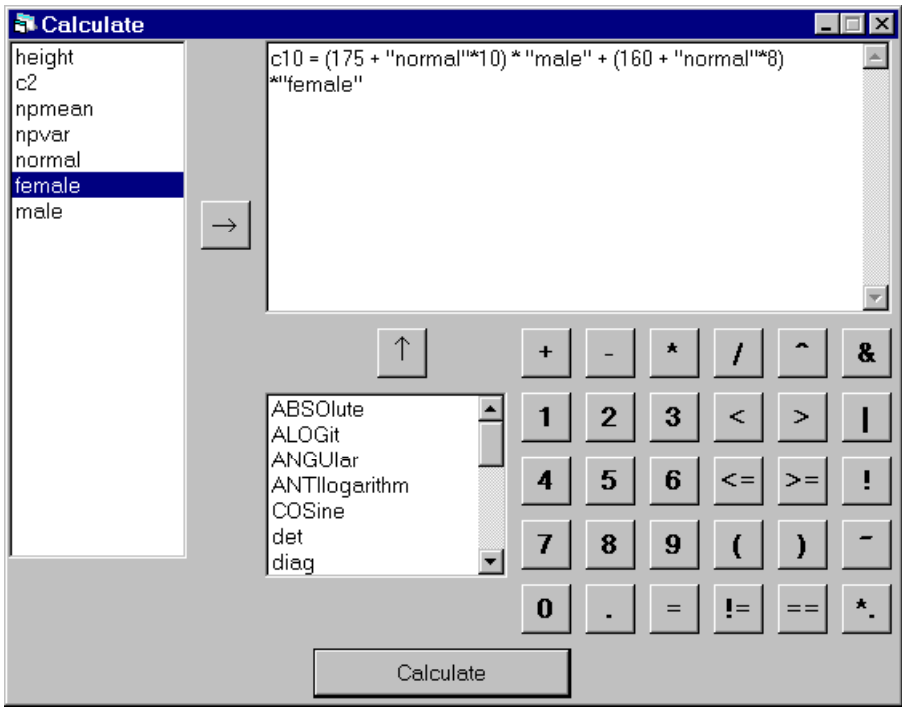
Clicking on **Generate** will now put the 100 random draws into column c7 (Incidentally this is equivalent to the NRAN command used in the parametric bootstrap macro). We now want to generate another 100 random numbers this time from a Binomial distribution to represent whether the person is male or female. To do this set up the **Generate Random Numbers** screen as shown below.



We can now name columns c7 'Normal' and c8 'Female' and use the **Calculate** window from the **Data Manipulation** menu to make another variable 'Male' in c9 that is equal to 1 – 'Female' as follows:

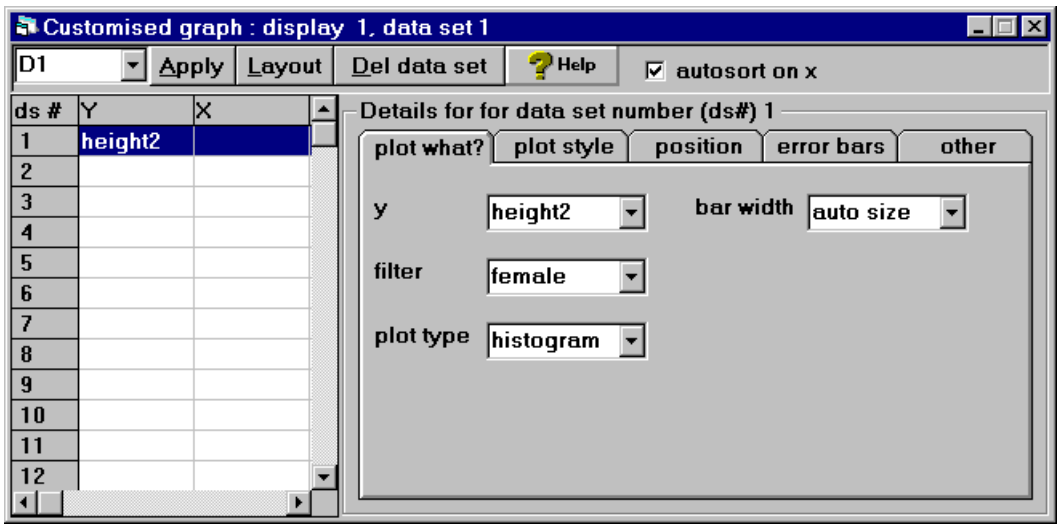


We can now at last construct the actual height variable (c10) as follows:

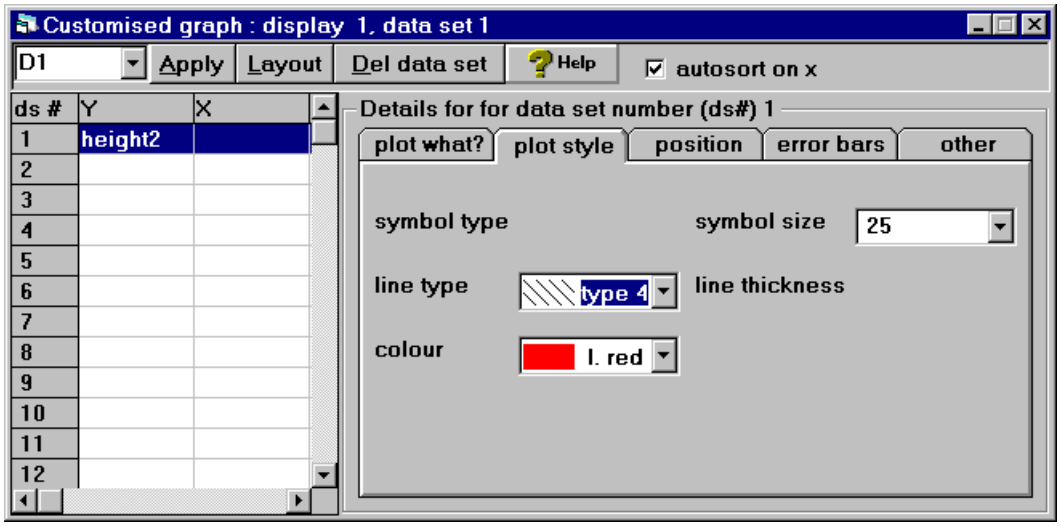


We will name column c10 'height2' using the **Names** window. We can now construct a histogram to look at our dataset. In this plot we will plot the males and females in different colours to show the make up of the mixture model.

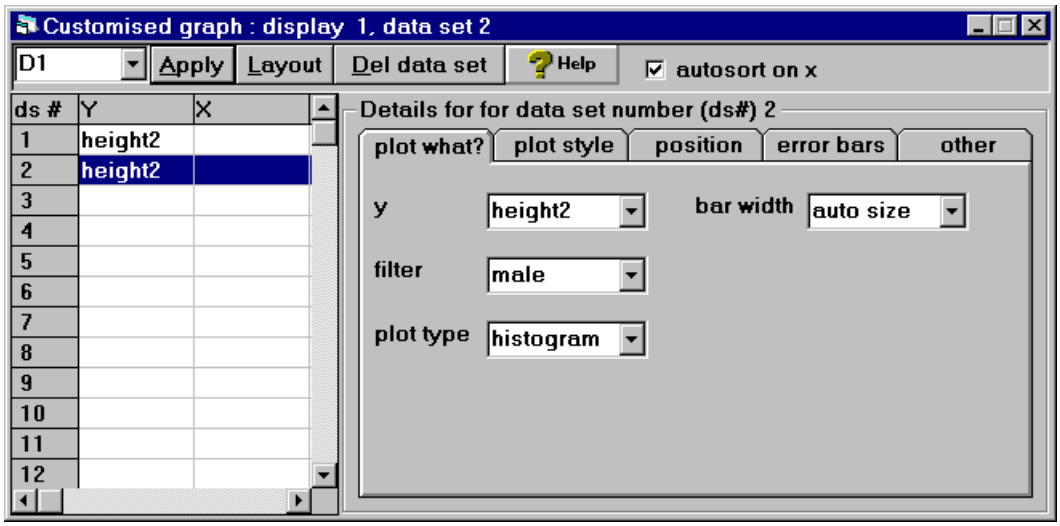
We will use display **D1** on the **Customised Graph(s)** option from the **Graphs** menu to construct the histogram. If you already have a graph set up on display **D1** then you can either delete any existing datasets or use a different display number. We will firstly set up the female heights. To do this select dataset 1 and set up the **plot what?** tab as follows



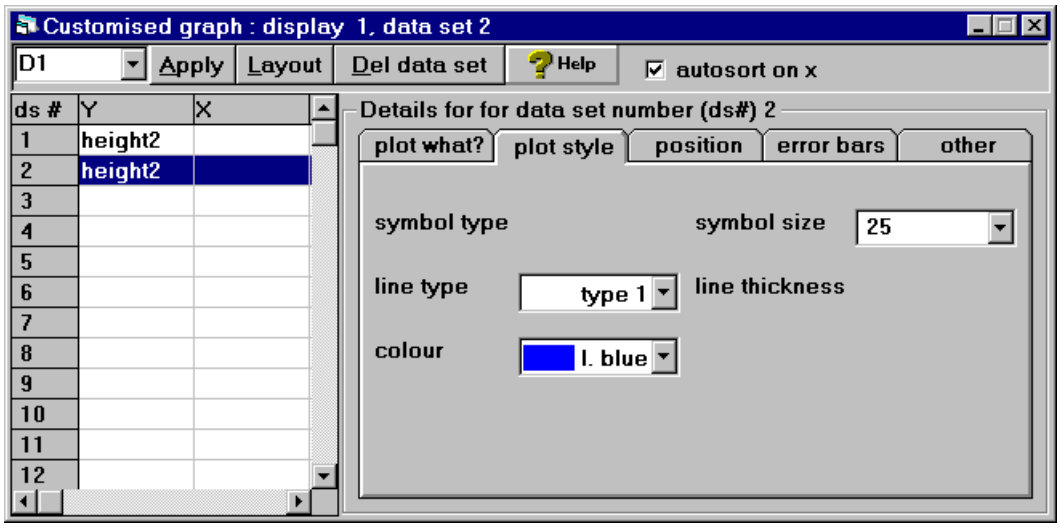
The filter option tells the software to only plot points when female is equal to 1. We also want to set different styles for the female and male heights so we set up **the plot style** tab as follows:



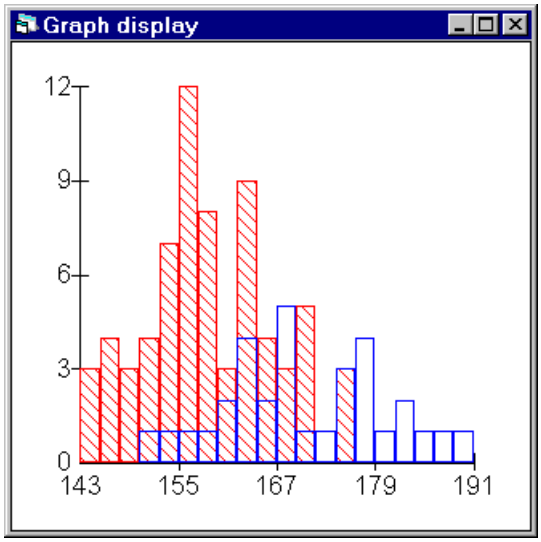
We now need to add the male heights to the graph. To do this we construct the male heights in dataset 2 that will be plotted on the same graph. The **plot what?** tab should be set as follows :



Again we need to set the **plot style** tab for this new graph. This should be done as follows:



Having finally set up the two datasets we can press the **Apply** button to view the histogram. The graph should now be displayed as follows:



This graph clearly shows the two distributions from which our heights have been generated. In this example we have the sex of each individual whose height was measured and so we could quite reasonably fit separate Normal models for males and females and construct interval estimates for each separately. It may however be the case that we do not know the sex of the individuals and here we could use a non-parametric bootstrapping method to fit the dataset.

Summary of simulation methods

In this chapter we have had a basic introduction to simulation methods through the use of the bootstrap on some simple examples. In the next chapter we will introduce the

other simulation methods used in *MLwiN*, MCMC methods and explain the theory behind their use. Then we will show to use the MCMC methods available in *MLwiN* in the next two chapters, firstly to fit Normal models and secondly to fit discrete response models. Finally we will look at how the bootstrapping methods introduced here can be modified to fit multilevel models in *MLwiN*

What you should have learnt from this chapter

- That simulation-based methods can be used as an alternative to the iterative methods used so far to fit multilevel models.
- How to use bootstrapping to fit simple models
- How to generate random numbers in *MLwiN*.
- How to write and execute macros in *MLwiN*.
- How to create histograms in *MLwiN*.

Chapter 14: Introduction to MCMC estimation

Bayesian modelling using Markov chain Monte Carlo methods

For Bayesian modelling *MLwiN* uses a combination of two Markov chain Monte Carlo (MCMC) procedures: Gibbs sampling and Metropolis-Hastings sampling. In release 1.1 of *MLwiN*, MCMC estimation is restricted to Normal models with a single variance at level 1 and generalised linear models with Binomial or Poisson errors.

We will start this chapter with some of the background and theory behind MCMC methods and Bayesian statistics before going on to consider the analysis of a dataset with a continuous Normal response. We will be using the same examination dataset described in Part 1 of this manual and fitting the variance components model studied there.

Users of *MLwiN* release 1.0 will find that the MCMC options and screen layouts have been modified in release 1.1 and may find this chapter (and the next two) useful to familiarise themselves with the new structure. The MCMC interface modifications are due to the addition of new features and enhancements, and the new interface is designed to be more intuitive.

MCMC methods and Bayesian modelling

The IGLS and RIGLS methods considered thus far in this manual find point estimates for the unknown parameters of interest in the model. These methods are based on iterative procedures and the process involves iterating until two consecutive estimates for each parameter are sufficiently close together and hence convergence has been achieved. These methods give point estimates for all parameters, estimates of the parameter standard deviations and large sample hypothesis tests and confidence intervals (see Part I). It will however also be useful to have methods for producing accurate interval estimates with small samples, and this is where MCMC estimation fits in (see also bootstrapping).

The examination dataset of Part I can be described by the following simple variance components model:

$$y_{ij} = \beta_{0ij} x_0 + \beta_1 x_{1ij}$$

$$\beta_{0ij} = \beta_0 + u_{0j} + e_{0ij}$$

$$u_{0j} \sim \mathrm{N}(0, \sigma_{u0}^2)$$

$e_{0ij} \sim \mathrm{N}(0, \sigma_{e0}^2)$

In a Bayesian formulation of this model we have the opportunity to combine *prior* information about the fixed and random parameters, β_0, β_1, σ_{u0}^2, σ_{e0}^2, with the data. These parameters are regarded as random variables described by probability distributions, and the prior information for a parameter is incorporated into the model via a *prior distribution*. After fitting the model, a distribution is produced for the above parameters that combines the prior information with the data is known as the *posterior distribution*.

When using MCMC methods we are now no longer aiming to find simple point estimates for the parameters of interest. Instead MCMC methods make a large number of simulated random draws from the joint posterior distribution of all the parameters, and use these random draws to form a summary of the underlying distribution. From the random draws of the parameter of interest, it is then possible to calculate the posterior mean and standard deviation (SD), as well as density plots of the complete posterior distribution and thus quantiles of this distribution.

In the rest of this section, the aim is to give users sufficient background material to have enough understanding of the concepts behind both Bayesian statistics and MCMC methods to allow them to use the MCMC options in the package. For the interested user, the book by Gilks, Richardson and Spiegelhalter (1996) gives more in-depth material on these topics than is covered here.

Default Prior Distributions

In Bayesian statistics, every unknown parameter **must** have a prior distribution. This distribution should describe all information known about the parameter prior to data collection. Often little is known about the parameters *a priori*, and so default prior distributions are required that express this lack of knowledge. The default priors applied in *MLwiN* when MCMC estimation is used are 'flat' or 'diffuse' for all the parameters. In *MLwiN* release 1.1 the following diffuse prior distributions are used (note these are slightly different from the default priors used in release 1.0):

- For fixed parameters $p(\beta) \propto 1$. This *improper* prior is functionally equivalent to a *proper* Normal prior with variance c^2, where c is extremely large with respect to the scale of the parameter. An *improper* prior distribution is a function that is not a true probability distribution in that it does not integrate to 1. For our purposes we only require the posterior distribution to be a true or *proper* distribution.

- For scalar variances, $p(1/\sigma^2) \sim Gamma(\varepsilon, \varepsilon)$, where ε is very small. This (proper) prior is more or less equivalent to a Uniform prior for $log(\sigma^2)$.

- For variance matrices $p(\Omega^{-1}) \sim \text{Wishart}_n(n, \hat{\Omega})$ where n is the number of rows in the variance matrix and $\hat{\Omega}$ is an estimate for the true value of Ω. The estimate $\hat{\Omega}$ will be the starting value of Ω (usually from the IGLS/RIGLS estimation routine) and so this prior is essentially an informative prior. However the first parameter, which represents the sample size on which our prior belief is based, is set to the smallest possible value (n the dimension of the variance matrix) so that this prior is only weakly informative.

These variance priors have been compared in Browne (1998), and some follow up work has been done on several different simulated datasets with the default priors used in release 1.0. These simulations compared the biases of the estimates produced when the true values of the parameters were known. It was shown that the new priors tend to generally give less biased estimates (when using the mean as the estimate) than the previous default priors used in release 1.0 although both methods give estimates with similar coverage properties. The priors used in release 1.0 and informative priors can also be specified and these will be discussed later in this chapter.

MCMC Estimation

The multilevel models fitted in *MLwiN* contain many unknown parameters of interest, and the objective of MCMC estimation of these models is to generate a sample of points in the space defined by the *joint posterior distribution* of these parameters. In the Normal variance components model (1.8) this consists of generating samples from the distribution $p(\beta_0, \beta_1, u_0, \sigma_{u0}^2, \sigma_{e0}^2 \mid y)$ where u_0 is the vector of u_{0j}'s.

Unfortunately to calculate this distribution directly will involve integrating over many parameters which in all but the simplest examples proves intractable, but fortunately an alternative approach is available. This is due to the fact that although the joint posterior distribution is difficult to simulate from, the *conditional posterior distributions* for the unknown parameters often have forms that can be simulated from easily.

Gibbs Sampling

The first MCMC method we will consider is *Gibbs Sampling*. Gibbs sampling works by simulating a new value for each parameter in turn from its conditional distribution assuming that the current values for the other parameters are the true values. For example consider again the Normal variance components model. We could split the parameters and level 2 residuals up into 4 subsets, β, u_0, σ_{u0}^2, and σ_{e0}^2 where $\beta = (\beta_0, \beta_1)$.

Note that, given the values of the fixed parameters and the level 2 residuals the level 1 residuals e_{0ij} can be calculated by subtraction and so are not included in the algorithms that follow.

We firstly need to choose starting values for each set of parameters, $\beta(0)$, $u_0(0)$, $\sigma_{u0}^2(0)$, and $\sigma_{e0}^2(0)$ and these are taken from the current values stored in *MLwiN* before MCMC estimation is started. For this reason it is **important** to run IGLS or RIGLS before running MCMC estimation to give the method good starting values. The method then works by sampling from the following conditional posterior distributions, firstly

$p(\beta \mid y, u_0(0), \sigma_{u0}^2(0), \sigma_{e0}^2(0))$ to generate $\beta(1)$, and then from

$p(u_0 \mid y, \beta(1), \sigma_{u0}^2(0), \sigma_{e0}^2(0))$ to generate $u_0(1)$, and then from

$p(\sigma_{u0}^2 \mid y, \beta(1), u_0(1), \sigma_{e0}^2(0))$ to generate $\sigma_{u0}^2(1)$, and then from

$p(\sigma_{e0}^2 \mid y, \beta(1), u_0(1), \sigma_{u0}^2(1))$ to generate $\sigma_{e0}^2(1)$.

Having performed all 4 steps we have now updated all of the unknown quantities in the model. This process is then simply repeated many times using the previously generated set of parameter values to generate the next set. The chain of values generated by this sampling procedure is known as a *Markov chain*, as every new value generated for a parameter only depends on its previous values through the last value generated.

This method works well if the conditional posterior distributions are easy to simulate from (which for Normal models they are) but this is not always the case. When the conditional posterior distributions do not have simple forms we will consider a second MCMC method, called *Metropolis Hastings* sampling. In general MCMC estimation methods generate new values from a *proposal distribution* that determines how to choose a new parameter value given the current parameter value. As the name suggests a proposal distribution suggests a new value for the parameter of interest. This new value is then either accepted as the next iteration or rejected and the current value is used as the next iteration. The Gibbs sampler has as its proposal distribution the conditional posterior distribution, and is a special case of the Metropolis Hastings sampler where every proposed value is accepted.

Metropolis Hastings Sampling

In general almost any distribution can be used as a proposal distribution. In *MLwiN*, the Metropolis Hastings sampler uses Normal proposal distributions centred at the current parameter value. This is known as a random-walk proposal. This proposal distribution, for parameter θ say, has the property that it is symmetric in $\theta(t\text{-}1)$ and $\theta(t)$, that is

$$p(\theta(t) = a \mid \theta(t\text{-}1) = b) = p(\theta(t) = b \mid \theta(t\text{-}1) = a),$$

and MCMC sampling with a symmetric proposal distribution is known as *pure Metropolis sampling*. The proposals are accepted or rejected in such a way that the

chain values are indeed sampled from the joint posterior distribution. As an example of how the method works the updating procedure for the parameter β_0 at time step t in the Normal variance components model is as follows:

- Draw β_0^* from the proposal distribution $\beta_0(t) \sim N(\beta_0(t\text{-}1), \sigma_p^2)$ where σ_p^2 is the proposal distribution variance.

- Define $r_t = p(\beta_0^*, \beta_1, u_0, \sigma_{u0}^2, \sigma_{e0}^2 \mid y) / p(\beta_0(t\text{-}1), \beta_1, u_0, \sigma_{u0}^2, \sigma_{e0}^2 \mid y)$ as the posterior ratio and let $a_t = \min(1, r_t)$ be the acceptance probability.

- Accept the proposal $\beta_0(t) = \beta_0^*$ with probability a_t, otherwise let $\beta_0(t) = \beta_0(t\text{-}1)$.

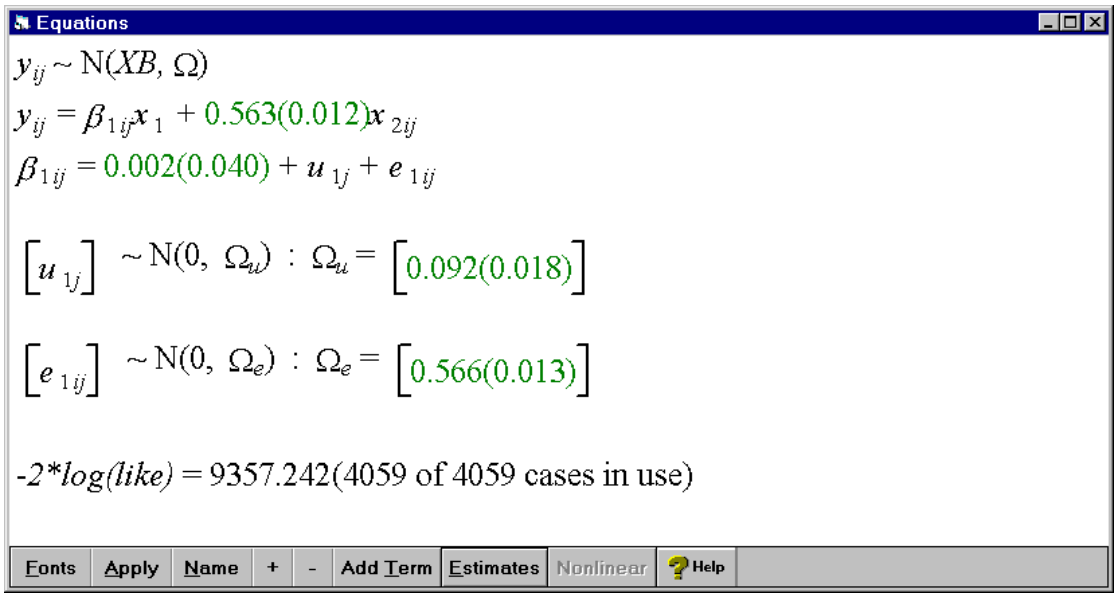
So from this algorithm you can see that the method either accepts the new value or rejects the new value and the chain stays where it is. The difficulty with Metropolis Hastings sampling is finding a 'good' proposal distribution that induces a chain with low autocorrelation. The problem is that, since the output of an MCMC algorithm is a realisation of a Markov chain, we are making (auto)correlated (rather than independent) draws from the posterior distribution. This autocorrelation tends to be positive, which can mean that the chain must be run for many thousands of iterations to produce accurate posterior summaries. When using the Normal proposals as above, reducing the autocorrelation to decrease the required number of iterations equates to finding a 'good' value for σ_p^2, the proposal distribution variance. We will see later in the examples the methods *MLwiN* uses to find a good value for σ_p^2.

As the Gibbs sampler is a special case of the Metropolis Hastings sampler, it is possible to combine the two algorithms so that some parameters are updated by Gibbs sampling and other parameters by Metropolis Hastings sampling as will be shown later. It is also possible to update parameters in groups by using a multivariate proposal distribution and this will also be demonstrated in the examples.

MCMC Estimation in *MLwiN*

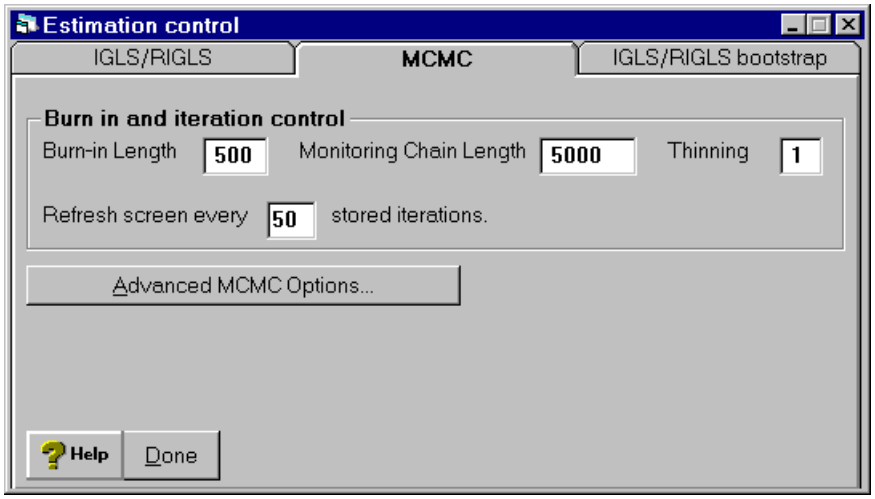
We will now consider how the theory discussed in the previous section works in practice by looking at an example of how to run MCMC sampling in *MLwiN*. For this we will consider the Normal response model example from Part I of this manual.

We retrieve the examination data worksheet tutorial.ws by selecting the **Open Worksheet** option from the **File** menu. Then we set up and run the variance components model using the IGLS method by pressing **Start** to get starting values for the MCMC method. We obtain on IGLS convergence the following results:



Gibbs Sampling

Now select **MCMC** by clicking on **Estimation control** and then **MCMC**. You will then see a screen similar to the following:



In release 1.1 of *MLwiN* the user does not have to choose between Gibbs sampling and Metropolis Hastings sampling directly. The software chooses the default (and most appropriate) technique for the given model, which in the case of Normal response models is Gibbs sampling for all parameters. The user can however modify the estimation methods used on the Advanced MCMC Options screen that will be discussed later.

The four boxes under the heading **Burn in and iteration control** have the following functions:

Burn-in Length. This is the number of initial iterations that will not be used to describe the final parameter distributions; that is they are discarded and used only to initialise the Markov chain. The default of 500 can be modified.

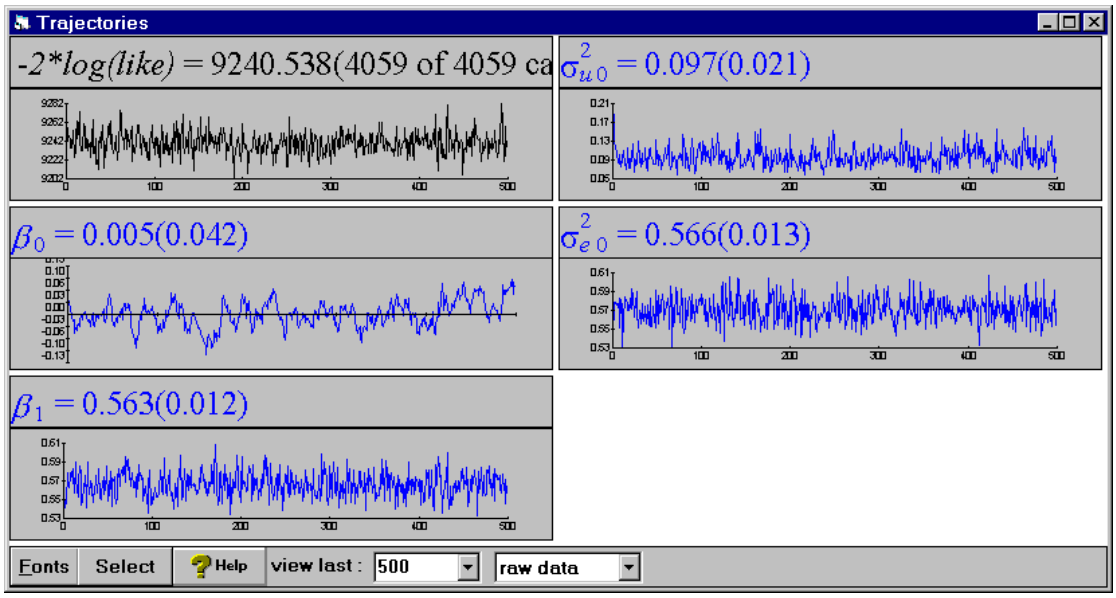
Monitoring Chain Length. The monitoring period is the number of iterations, after the burn-in period, for which the chain is to be run. The default of 5000 can be modified. Distributional summaries for the parameters can be produced either at the end of the monitoring run or at any intermediate time.

Thinning. This is the frequency with which successive values in the Markov chain are stored. This works in a more intuitive way in release 1.1, for example running a chain for a monitoring chain length of 50,000 and setting thinning to 10 will result in 5,000 values being stored. The default value of 1, which can be changed, means that every iteration is stored. The main reason to use thinning is if the monitoring run is very long and there is limited memory available. In this situation this parameter can be set to a higher integer, *k*, so that only every *k*-th iteration will be stored. Note, however, that the parameter mean and standard deviation use all the iteration values, no matter what thinning factor is used. All other summary statistics and plots are based on the thinned chain only.

Refresh. This specifies how frequently the parameter estimates are refreshed on the screen during the monitoring run within the equations and trajectories windows. The default of 50 can be changed.

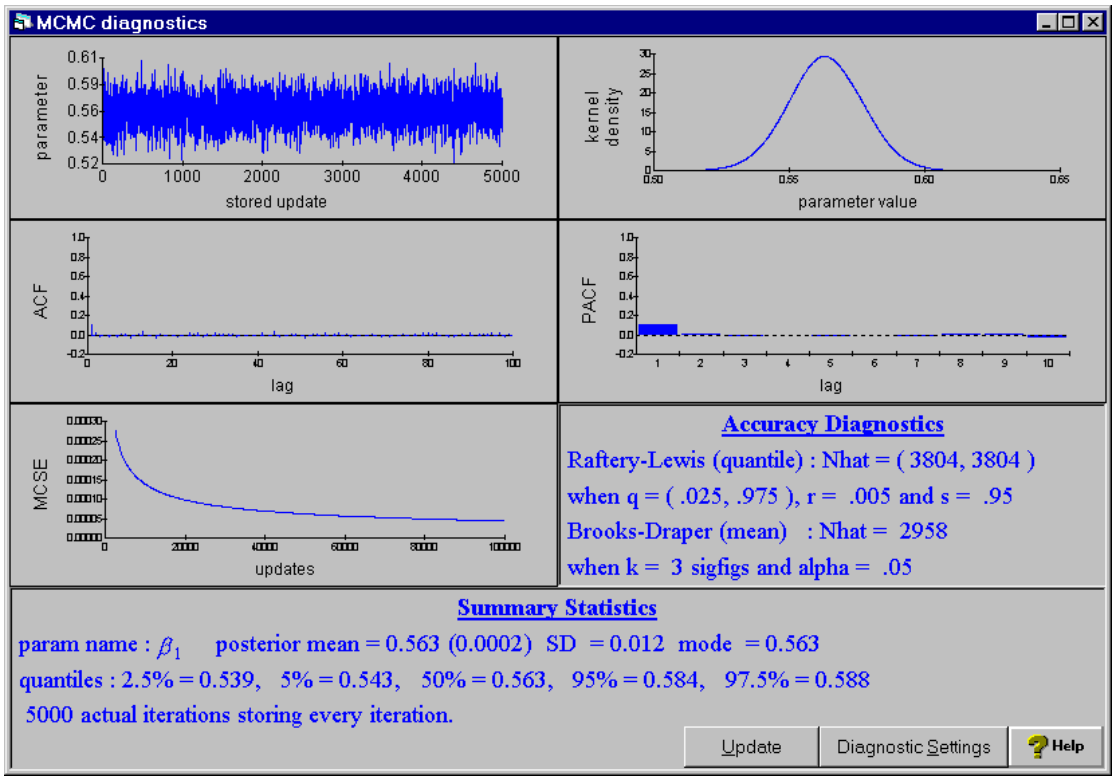
Running the Gibbs Sampler

We will use the default values on the Estimation Control window, so click on **Done** or close down the window. Then open both the **Trajectories** window and the **Equations** window from the **Model menu** and reposition the windows so that both can be seen. Now click on the **Start** button and the words **BURNING IN** will appear for the duration of the burn in period. After this the iteration counter at the bottom of the screen will move every 50 iterations and both the Equations and Trajectories windows will show the current parameter estimates (based on the chain means) and standard deviations. After the chain has run for 5,000 iterations the trajectories window should look similar to the following:



Healthy Gibbs sampling traces should look like the iteration trace for β_1; when considered as a time series these traces should resemble 'white noise'. At the bottom of the screen you will see two default settings. The first allows you to choose how many values to view and here shows the values for the previous 500 iterations only; this can be changed. The second drop down menu allows you to switch from showing the actual chain values to viewing the running mean of each parameter over time.

To get more detailed diagnostic information about a parameter, for example the slope coefficient β_1, click the left mouse button on the parameter trace for β_1. The program will then ask 'Calculate MCMC diagnostics?' to which you should click on 'Yes'. The message 'Calculating MCMC diagnostics … May take a while.' will then appear and after a short wait you will see a diagnostics screen similar to the following:



The upper left-hand cell simply reproduces the whole trace for the parameter. The upper right-hand cell gives a kernel density estimate of the posterior distribution; when an *informative* prior distribution is used the density for this distribution is also displayed in black (see example later). We can see in this example that the density looks to have approximately a Normal distribution. The second row of boxes plot the autocorrelation (ACF) and partial autocorrelation (PACF) functions. The PACF has a small spike at lag 1 indicating that Gibbs sampling here behaves like a first order autoregressive time series with a small autocorrelation of about 0.1. The ACF is consistent with this suggesting that the chain is adequately close to independently identically distributed (IID) data (autocorrelation 0).

The third row consists of some accuracy diagnostics. The left-hand box plots the estimated Monte Carlo standard error (MCSE) of the posterior estimate of the mean against the number of iterations. This allows the user to calculate how long to run the chain to achieve a mean estimate with a particular desired MCSE. The right-hand box contains two contrasting accuracy diagnostics. The Raftery-Lewis diagnostic (Raftery and Lewis 1992) is a diagnostic based on a particular quantile of the distribution. The diagnostic Nhat is used to estimate the length of Markov chain required to estimate a particular quantile to a given accuracy. In *MLwiN* the diagnostic is calculated for the two quantiles (the default is the 2.5% and 97.5% quantiles) that will form a central interval estimate. For this parameter the estimated chain length (Nhat) is 3,804 for both quantiles (note this is unusual and generally the quantiles will have different Nhat values) so having run the chain for 5,000 iterations we have satisfied this diagnostic. The Brooks-Draper diagnostic is a diagnostic based on the mean of the distribution. It

is used to estimate the length of Markov chain required to produce a mean estimate to *k* significant figures with a given accuracy. Here we can see that to quote our estimate as 0.563 (3 significant figures) with the desired accuracy requires the chain to be run for 2,958 iterations so this diagnostic is also satisfied for this parameter.

The interpretation of the numbers q = (0.025,0.975), r = 0.005 and s = 0.95 in the Raftery-Lewis diagnostic is as follows: With these choices the actual Monte Carlo coverage of the nominal 100(0.975 – 0.025)% = 95% interval estimate for the given parameter should differ by no more than 100(2*r)% = 1 percentage point with Monte Carlo probability 100*s = 95%. The values of q, r and s can be changed.

The bottom box contains some numerical summaries of the data. As well as the mean (with its MCSE in parenthesis), this box also contains the mode and median estimates. To estimate both 90% and 95% intervals this box also contains the appropriate quantiles of the distribution. For example a 95% central interval (Bayesian credible interval) runs from 0.539 to 0.588.

Note that many of the settings on the diagnostics screen can be changed from their default values. For more information on changing the diagnostics screen settings see the on-line Help system.

To see a somewhat different picture you can shut down this window and click, for example, on the plot for the level 2 variance in the **Trajectories window**. Finally, we can compare the results from Gibbs to the results from the RIGLS method for this model in the following table:

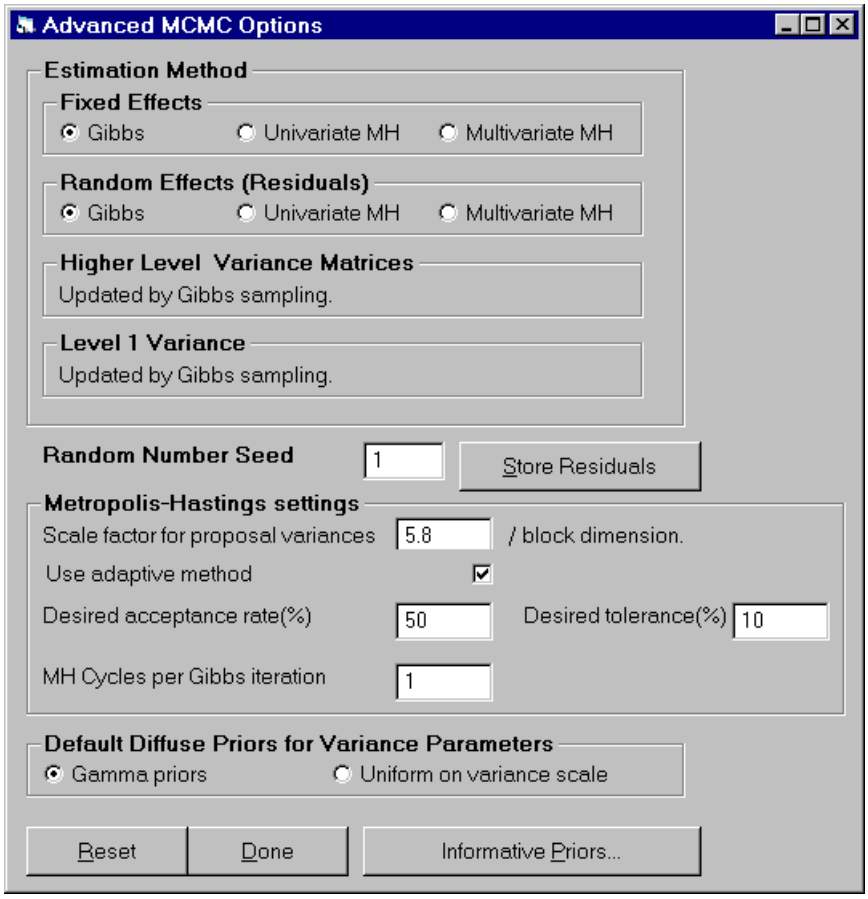
Parameter	Gibbs posterior		RIGLS	
	Mean	SD	Mean	SD
β_0	0.005	0.042	0.002	0.040
β_1	0.563	0.012	0.563	0.012
σ_{u0}^2	0.097	0.021	0.092	0.018
σ_{e0}^2	0.566	0.013	0.566	0.013

The only real difference is the slightly higher values for the Gibbs estimate of the level 2 variance. The Gibbs estimate for the mode of 0.092 is identical to the RIGLS estimate (to 3 decimal places) since the maximum likelihood estimate approximates (in effect)

the posterior mode (with a diffuse prior) rather than the mean. In some situations, the choice of diffuse prior (for the variance parameter) will be important, in particular when the underlying variance is close to zero and poorly estimated (i.e. with a large standard error). This may be particularly noticeable in random coefficient models and is a topic of current research (Browne 1998).

Metropolis Hastings (MH) Sampling

Although for Normal response models the default MCMC method is Gibbs sampling for all parameters, we can still use other methods via the **Advanced MCMC Options** window. Metropolis Hastings sampling is particularly useful for multilevel generalised linear models as will be seen in the other examples in this chapter. We shall firstly see how it can be used with the same examination dataset we used in Chapter 2. To use MH sampling, go back to the **Estimation control** window and click on **Advanced MCMC Options** which will bring up the following window:



This window contains the options to change the estimation methods used for various groups of parameters as well as other advanced options including a selection of Metropolis Hastings settings. As the window shows you can change the MCMC method used for both the fixed effect parameters and the residuals; the variance parameters are

always updated by the Gibbs sampling method. To use Metropolis Hastings sampling select the method Univariate MH for both the fixed effect parameters and the residuals. We now look at how to use the settings.

Metropolis-Hastings settings

The Metropolis Hastings sampler has some additional settings to help *MLwiN* choose 'good' proposal distributions for the parameters. There are two strategies that can be used in *MLwiN* to produce good proposal distributions. Firstly it was shown in Gelman, Roberts and Gilks (1995) that for a simple Normal posterior distribution, a univariate Normal proposal distribution with a variance 5.8 times the true variance of the parameter is the best proposal distribution. Hence in *MLwiN* the user has the option to input a scale factor for proposal variances. This number will then be multiplied by the estimated parameter variance (from IGLS/RIGLS) to give the proposal distribution variance. Although this works for single level Normal models, studies of multilevel models (Browne 1998) have shown that the factor 5.8 is not always the best and is often too high.

The second approach that is used by default in release 1.1 of *MLwiN* is to find proposal distributions that give a desired acceptance rate. Experience suggests that rates of between 30% and 70% provide a useful compromise between a proposal variance that is too large and a variance that is too small. If the proposal variance is too large, the chain stays where it is for long periods before making a large jump, where as if it is too small the chain makes lots of little moves but takes a long time to explore the whole sample space. *MLwiN* finds the desired proposal distribution by running an adapting period before the burn in. In this adapting period the proposal distribution is modified to improve the acceptance rate.

The settings screen contains two boxes labelled *desired acceptance rate* (default 50%) and *desired tolerance* (default 10%). If the adaptive method is selected then when you click **Start** *MLwiN* will make an exploratory run of up to 5,000 iterations while displaying the message 'Running Adaptive procedure and Burning in'. During this period the proposal variance is changed adaptively to ensure that the acceptance rates for all parameters are as close to 50% as possible, and in the range 50%-10%=40% to 50%+10%=60%. Once this is achieved the adapting period ends and the burn in begins as with Gibbs sampling.

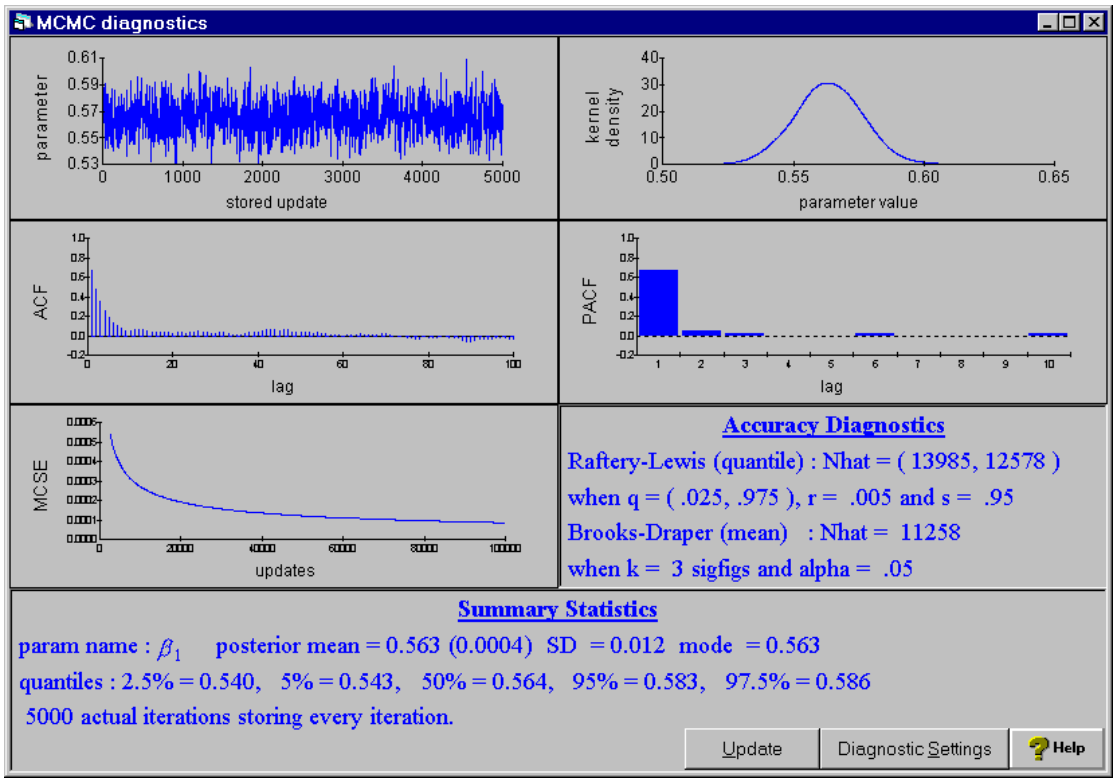
We will now look at running the examination dataset with the adaptive method.

Running the examination data with Metropolis Hastings

After setting up the variance components model as before and running the IGLS method

to get starting values, the Metropolis Hastings sampler was run with the default settings.

Note: to use Metropolis Hastings sampling select the method Univariate MH for both the fixed effect parameters and the residuals from the **Advanced MCMC Options** window. After 5,000 iterations clicking on the graph for β_1 in the **Trajectories** window you should now see diagnostics for β_1 similar to the following:



We can now see that both accuracy diagnostics give Nhat values that are considerably higher than for Gibbs sampling so that for this model MH would take longer than Gibbs to give the same accuracy. We also see that the first order autocorrelation is about 0.65 which is substantially higher than for Gibbs and also implies a longer chain length is needed. If we run the MH sampler without the adaptive method, we obtain a slightly higher Nhat value of about 17,000 for the Raftery-Lewis diagnostic for the 2.5% quantile.

The following table compares parameter estimates from RIGLS, Gibbs and MH with both the scale factor and adaptive methods.

Parameter	RIGLS	Gibbs	MH scale factor = 5.8	MH adaptive with defaults
β_0	0.002	0.005	0.010	-0.005
β_1	0.563	0.563	0.563	0.563
σ_{u0}^2	0.092	0.097	0.097	0.097
σ_{e0}^2	0.566	0.566	0.566	0.566

MH and Gibbs show good agreement, and apart from the level 2 variance parameter there is good agreement with RIGLS too. The estimates of the intercept β_0 show some variability but this is because this parameter has larger Nhat values and so the chains have not been run for long enough. It is often useful with MCMC estimation to try both Gibbs and MH to ensure stable conclusions.

In the next chapter we will go on to consider other more advanced features of the MCMC method as implemented in *MLwiN*.

What you should have learnt from this chapter

An introduction to MCMC methods.

The difference between Gibbs sampling and Metropolis Hastings sampling.

How to get estimates from MCMC methods and how to tell when an MCMC chain has converged.

An introduction to prior distributions.

Chapter 15: Advanced MCMC Sampling

In the last chapter we learnt how to use the MCMC methods in *MLwiN* to fit a simple multilevel model. In this chapter we will consider some other aspects of MCMC sampling in *MLwiN*, including new features that have been introduced in release 1.1 of *MLwiN*. This chapter follows directly on from the last chapter and continues to use the variance components model for the examination dataset.

New Metropolis Hastings Options

At the end of the last chapter we considered using the Metropolis Hastings sampler. In release 1.1 of *MLwiN* a few extra MH sampling options have been included which will be discussed here.

MH Cycles per Gibbs iteration

The parameter *MH cycles per Gibbs iteration* governs how many times the steps for the parameters being estimated by MH are run for each iteration. This is useful as the MH method tends to give higher autocorrelations than the Gibbs sampling method. For example if the parameter is set to 3 in the above example, the Raftery Lewis Nhat for β_1 reduced to 7,266 and the autocorrelation of its chain is reduced to 0.4. Note that this reduction in autocorrelation has to be balanced by the increase in time to run the model.

Block Updating MH Sampling

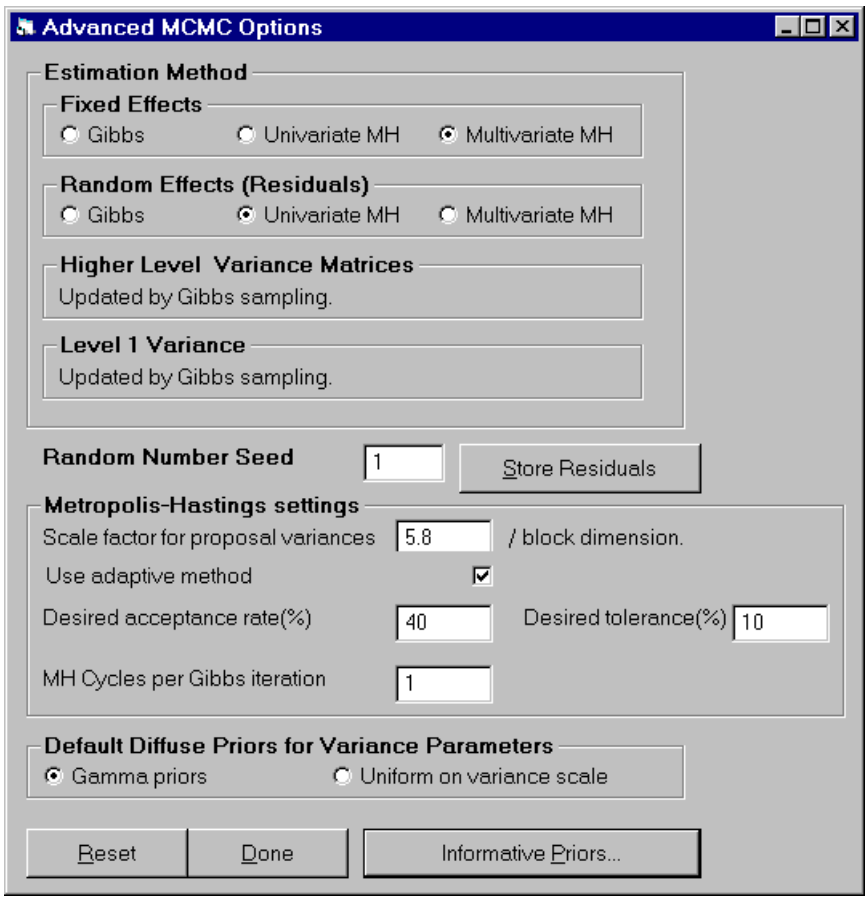
The Gibbs sampling estimation method in *MLwiN* updates the parameters in blocks; all fixed effects are updated together as are all the level 2 residuals for one level 2 unit. In contrast the MH estimation method uses univariate updates for each parameter separately. In release 1.1, a new 'experimental' block updating MH method has been included. This method updates the parameters in the same blocks used by the Gibbs sampling method.

Parameters are updated in blocks using multivariate Normal proposals that take account of the correlation between the parameters in the block. As several parameters are updated together acceptance rates for each block will be lower than the acceptance rates achieved by updating each parameter individually, although updating the block should be faster than updating each parameter individually.

Block Updating Example

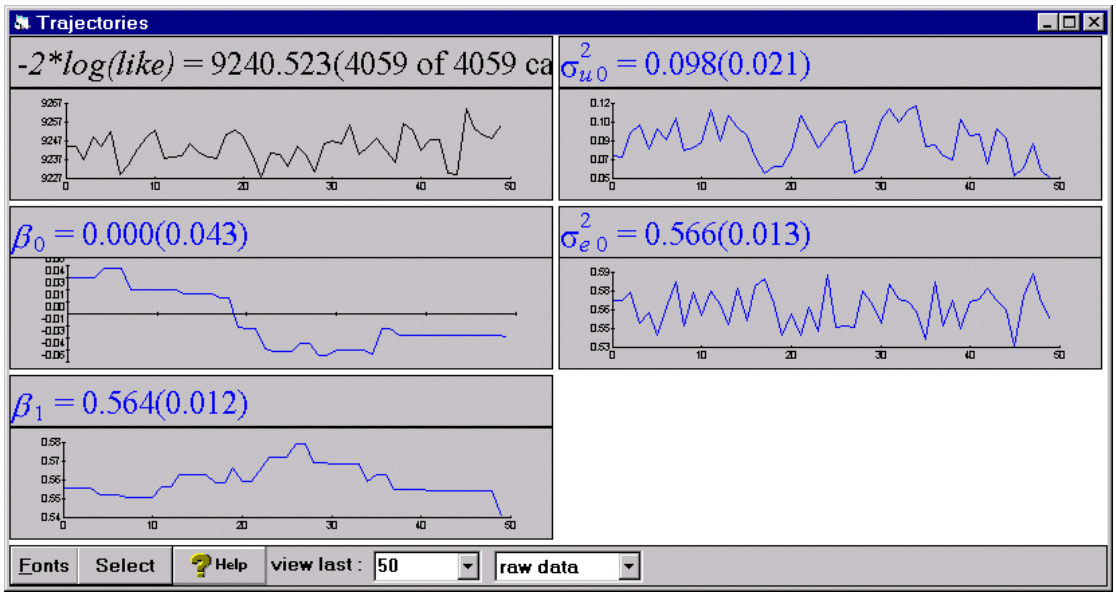
In the variance components example, there is only one set of level 2 residuals but 2 fixed effects. Consequently changing the estimation method to Multivariate MH i.e. block updating will have no effect for the residuals but will have an effect for the fixed

effects. After running IGLS, bring up the Advanced MCMC options window and change the settings as follows:



Note that in the Metropolis Hastings settings box we have now changed the desired acceptance rate to 40%. This is because the optimal acceptance rate for a block update is smaller the larger the block size. One of the reasons that the block updating method is regarded as 'experimental' is that ideally the optimal acceptance rate for the fixed effects and the optimal acceptance rate for the level 2 residuals will be different when the block sizes are different. In release 1.1 however the user may only enter a global desired acceptance rate. This is an area of active research (Browne 1998) and this may change in later releases of *MLwiN*.

The method was run for 5,000 iterations after an adapting period and trajectory traces of the last 50 iterations of the chains can be seen below:



On close inspection it can be seen that the two chains for the fixed effects, β_0 and β_1 are being updated as a block.

Prior Distributions in *MLwiN*

As mentioned in the introduction, the default priors for the variance parameters have now been changed from the defaults used in release 1.0. This is because these new priors generally give less positive bias when the estimate based on the mean is used. The old default priors can still be selected via the Advanced MCMC Options window as can informative priors.

Uniform on Variance scale priors

The default variance priors used in *MLwiN* (release 1.0) are now offered as an alternative to the new default priors. The *improper* diffuse priors used previously were as follows:

- For random parameters priors are placed on variances and covariance matrices

  $p(\Omega) \propto 1$ (constant on the scale of Ω, or σ^2 for a single variance)

These priors are functionally equivalent to the following proper priors:

- For variance parameters, a Uniform prior $U(0,c)$ where c is chosen so that $(0,c)$ spans the range in which the likelihood for the parameter is non-negligible.

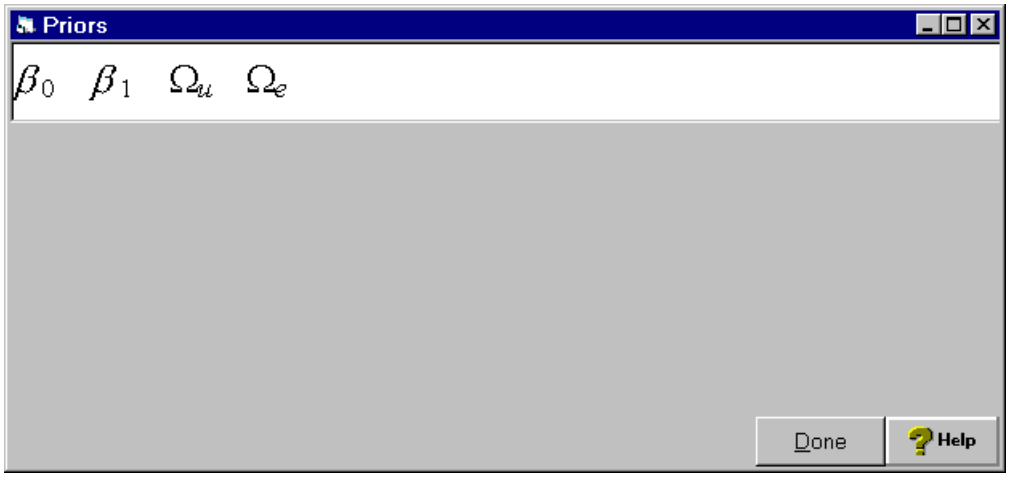
Comparing these priors using the Gibbs sampler on the variance components model we get the following results:

Parameter	RIGLS	Gibbs (Gamma prior)	Gibbs (Uniform prior)
β_0	0.002 (0.040)	0.005 (0.037)	0.004 (0.042)
β_1	0.563 (0.012)	0.563 (0.013)	0.563 (0.013)
σ_{u0}^2 (Mean) σ_{u0}^2 (Mode)	0.092 (0.018) -	0.097 (0.020) 0.092	0.101 (0.022) 0.095
σ_{e0}^2	0.566 (0.013)	0.566 (0.013)	0.566 (0.013)

An MCMC example with informative priors

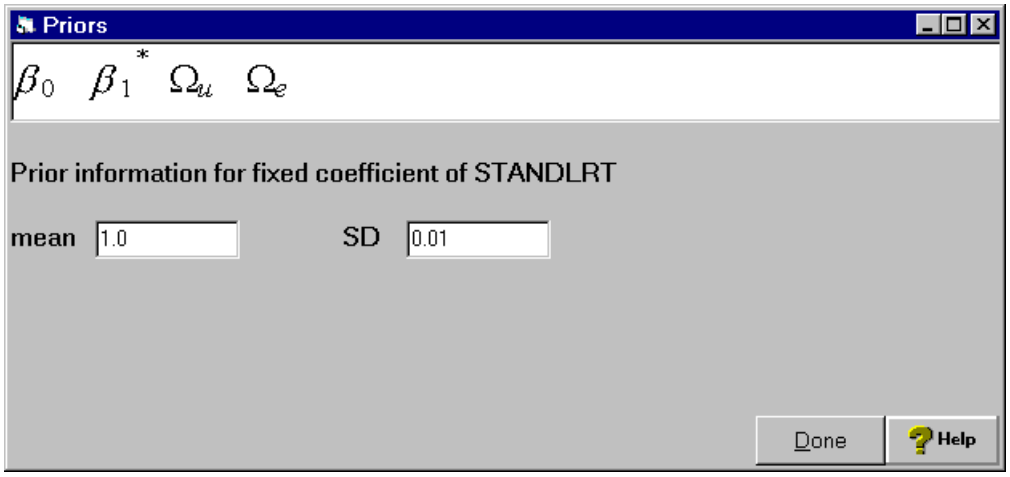
In this example we shall see the effect of using informative priors with Gibbs sampling.

Again we will start with IGLS starting values so run the model using the IGLS method. On the *Advanced MCMC Options* screen click on **Reset** to set the options back to Gibbs sampling for all parameters. Now click on the **Informative Priors** button and the following window will appear showing all the parameters in the model (the level 1 and level 2 covariance matrices in fact in this case just contain a single variance term).



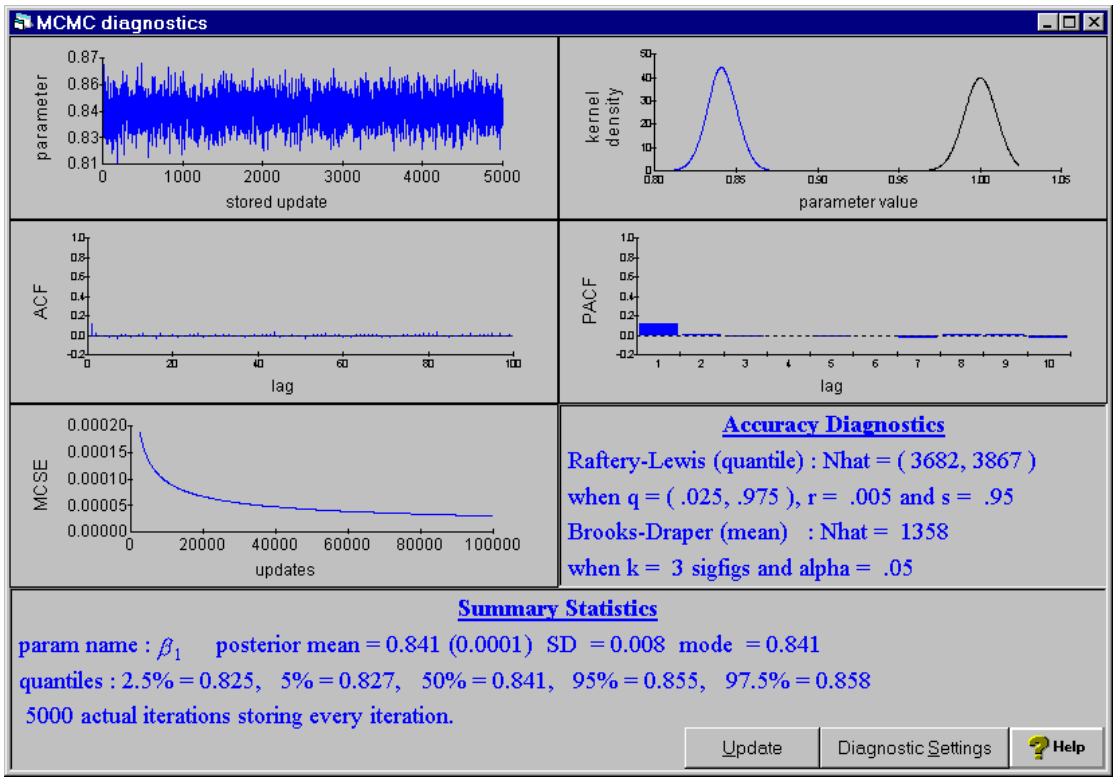
For the fixed parameters, priors are assumed to be Normal and are chosen by specifying the mean and SD. The priors for a covariance matrix are assumed to have an inverse

Wishart distribution and the specification is described in the next section. If you click on, for example, β_1, and then enter an informative prior with a mean of 1 (remembering the estimate is just over half this) and a prior SD of 0.01 (implying highly accurate prior knowledge), and click on β_1 again the window will appear as follows:



The asterisk that appears next to β_1 indicates that a prior distribution has been set for this parameter. (You can get back to the default by clearing the values and clicking next to the asterisk.) Now click on **Done** to close down the priors window and click start to run the Gibbs sampler.

After 5000 iterations, if you click on the trajectories window for β_1, the diagnostics plot will appear similar to that below:

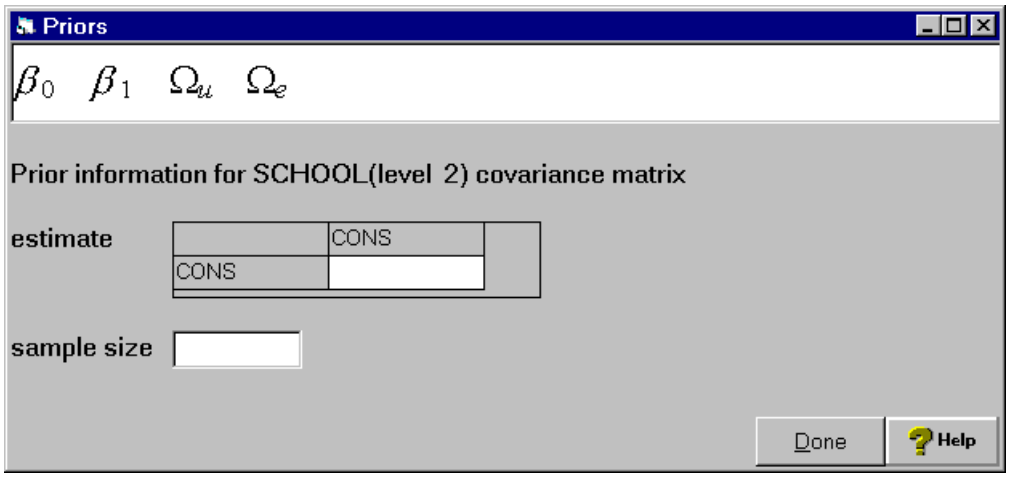


Here we can see that the prior distribution (on the right) is included on the kernel density plot in black and can thus be compared with the posterior distribution, which is in blue. In this example the two distributions are completely separated, and of roughly equal information content, indicating a conflict between the prior and the data. The estimate of the parameter is also very different from what we had before (0.841 as opposed to 0.563) and the variance parameter estimates are also rather different, which reduces the SD for β_1.

Now let us specify less accuracy for the prior by changing the value of SD to 1 and running the model again. This time we obtain values for all parameters that are very close to those of the default prior assumptions, because the prior is now highly diffuse in relation to the data. Note that changing the structure of the current model will result in all informative prior information being erased.

Specifying an informative prior for a random parameter

The procedure for specifying informative priors for random parameters is somewhat different. Click on the **STOP** button to end the previous estimation. Clear the existing prior on the slope coefficient and then click on the level 2 variance matrix Ω_u, in the priors window and you will see the following:



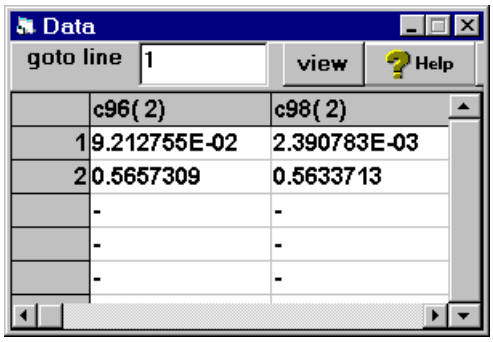
For our prior estimate of the level 2 variance let us choose 0.2 (the RIGLS estimate is 0.092). The sample size indicates the precision of this estimate. Thus, for example, if you had an estimate from another study based on 100 schools, you would enter the value 100. Let us do this, remembering that there are 65 schools in the present study. The estimated value of the level 2 variance after 5,000 iterations is now 0.163 – close to a weighted average of the estimate obtained with a diffuse prior and the informative prior estimate, and the other parameter estimates are hardly changed. In release 1.1 of *MLwiN* the prior density is not shown for random parameters.

Changing random number seed and parameter starting values

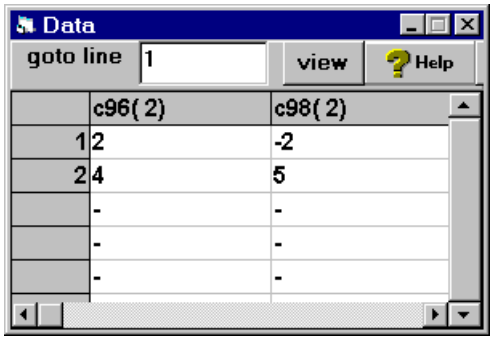
MCMC sampling involves making random draws from the conditional posterior distributions of the various parameters in the multilevel model. To start the MCMC sampler, starting values are required for each parameter along with a starting value (a positive integer) for the random number generator known as a **seed**. This means that given the starting values of the parameters and the random number **seed** the sampling is deterministic and so running the same model with the same starting values and seed on a different machine will give the same answers.

Most of the posterior distributions that can be run using MCMC in *MLwiN* are known to be uni-modal. If this were not the case then it would be more sensible to make several runs with different starting values to check that they all reached the same final estimates. The starting values for the fixed effects and variance parameters are taken from the values obtained by the last model run. These values are stored on the worksheet in columns c96 and c98 respectively.

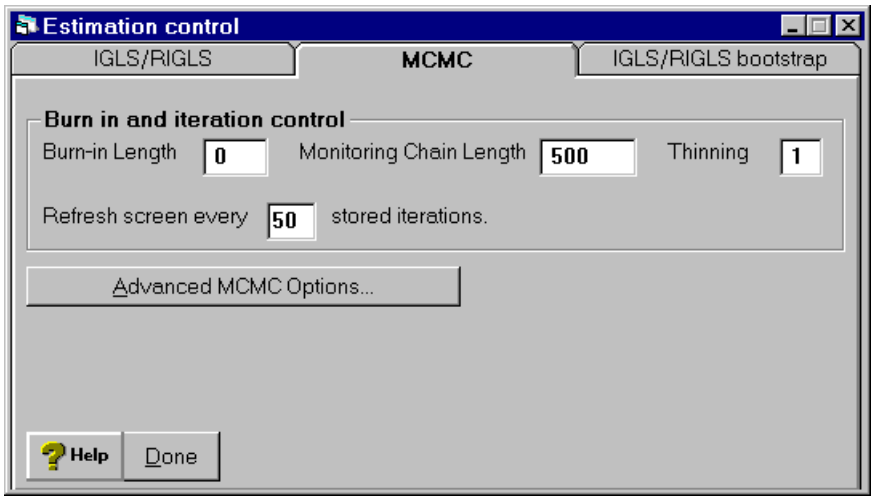
We will consider again our 2 level variance components model and run IGLS on the model. If you then open the **Data** window and select to view only columns c96 and c98 you should see the following:



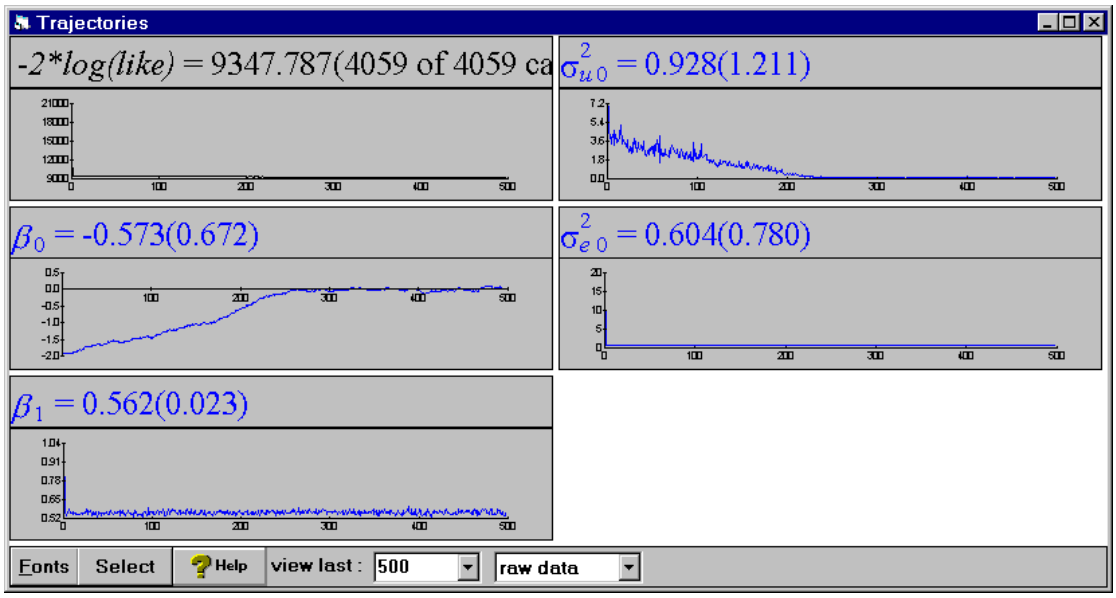
We will now alter these estimates to values that are far less plausible. To alter a value in the **Data** window**,** simply click on a cell in the window and type the new value. The new values were chosen as follows:



Note that *MLwiN* uses the starting values of the other parameters to calculate the starting values of the residuals, and so these new starting values cannot be altered directly. To see the progress of the chains from their starting values we will set the burn-in length to 0 and the monitoring chain length to 500 in the **Estimation Control** window as shown below.



The chains for the first 500 iterations can be seen in the following **Trajectories** window:



By about 250 iterations all the parameters appear to settle out at roughly the same estimates as seen when using the IGLS starting values. This means that if we had set a burn-in length of 500 iterations we would not have even seen this behaviour! If you now run for another 500 iterations the trajectories plots of the second 500 iterations will look similar to the Gibbs chains using the IGLS starting values.

Running from different starting values is useful for checking that all the parameters in your model have unimodal posterior distributions. If this is already known it is best to utilise the 'good' starting values obtained by IGLS, particularly when using MH sampling which may take longer to reach the point where it is actually sampling from the correct posterior distribution.

The random number seed can be set on the advanced MCMC options screen. Changing the random number seed is another technique that can be used to check that the MCMC method is sampling correctly from the posterior distribution. Running from the same starting values but with different random number seeds will give different estimates but these estimates will hopefully be similar. To illustrate this behaviour the following table contains the point estimates, after 5000 iterations obtained for the variance components model using the Gibbs sampler with random number seeds 1 to 4.

Parameter	Seed 1	Seed 2	Seed 3	Seed 4
β_0	0.005	0.003	0.002	0.004
β_1	0.563	0.563	0.563	0.564
σ_{u0}^2	0.097	0.097	0.097	0.097
σ_{e0}^2	0.566	0.566	0.566	0.566

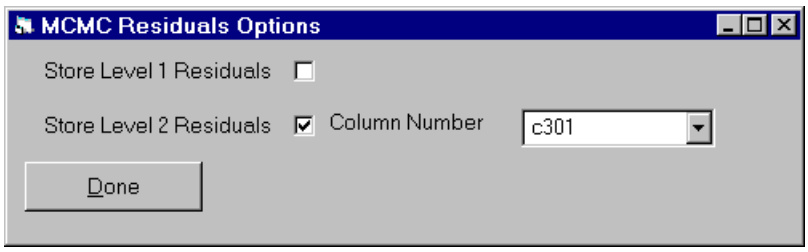
This table clearly shows that there is little change in the parameter values when we change the random number seed. This means we can have more confidence in our estimates.

Note that the options in this section are not generally required to ensure good MCMC performance. We include them only for completeness.

Residuals in MCMC

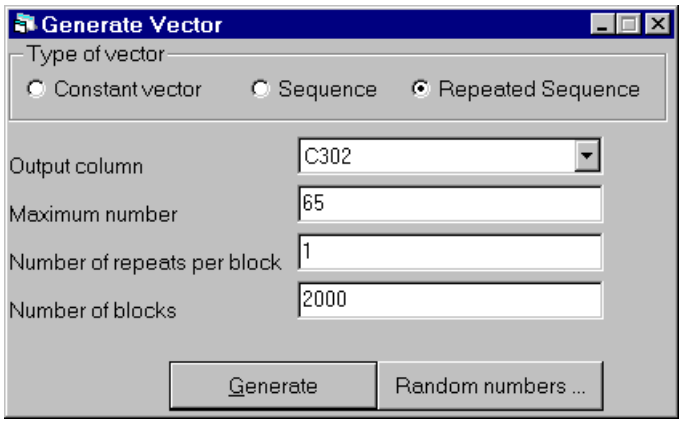
Although a multilevel model contains many parameters, by default when running MCMC sampling, the full MCMC chains are only stored for the fixed effects and variance parameters. For the residuals, means and standard errors are stored from their chains. It is then these values that are used when the residuals options as demonstrated in chapter 2 are used whilst running MCMC. If however an accurate interval estimate or other summary statistics are required for a residual then there is also the option of storing all the residuals at a given level.

To store residuals will generally require a large amount of memory as there are generally a large number of residuals per level. We will consider storing only the level 2 residuals here, although even then we will have 65 residuals, one per school. To store residuals click on the **Store Residuals** button on the **Advanced MCMC options** window and the **MCMC Residuals Options** window will appear. Click in the box to **Store Level 2 Residuals** and the window will look as follows:

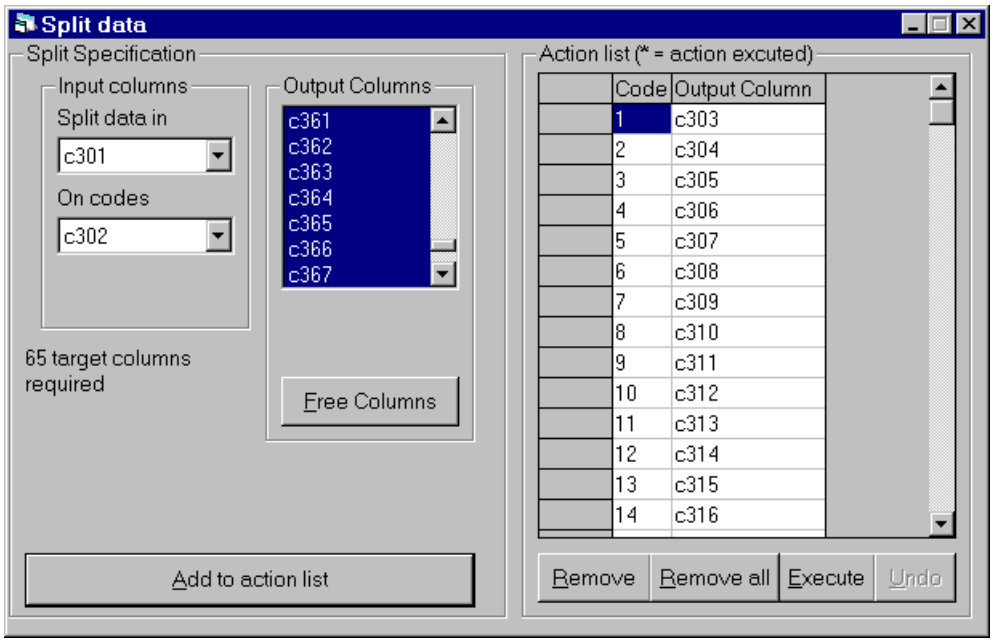


This option means that the chain values for the 65 residuals will be stacked in column 301. To avoid having to increase the worksheet size we will only run for 2,000 iterations. Click on **Done** and then run the model using Gibbs sampling as before except with a monitoring run length of 2,000. After running the model we will now have to

split column 301 into 65 separate columns, one for each residual. To do this we need to generate an indicator column that is a repeated sequence of the numbers 1 to 65 to identify the residuals. To generate this sequence of numbers in column c302 select **the Generate vector** window from the **Data Manipulation** menu and choose the options as shown in the window below.



Click on **Generate** to create the column. We now need to use the **Split column** option from the **Data Manipulation** menu to identify the columns that will contain the individual residual chains. From the Split data window select c301 as the data column and c302 as the code column. We will select c303 to c367 as the output columns. Here you will need to use the Shift key or the Ctrl key with the left mouse button to select multiple columns. Clicking on **Add to action list** will produce the following screen



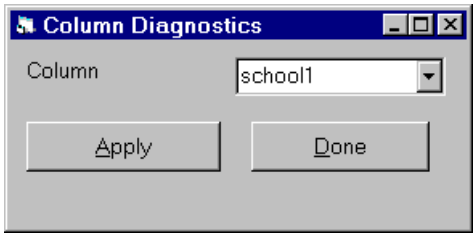
Clicking on **Execute** will now run the command and the columns c303-c367 will now contain the chains of the school level residuals. We could now erase the stacked data to

save space by typing in the **Command interface** window the following command

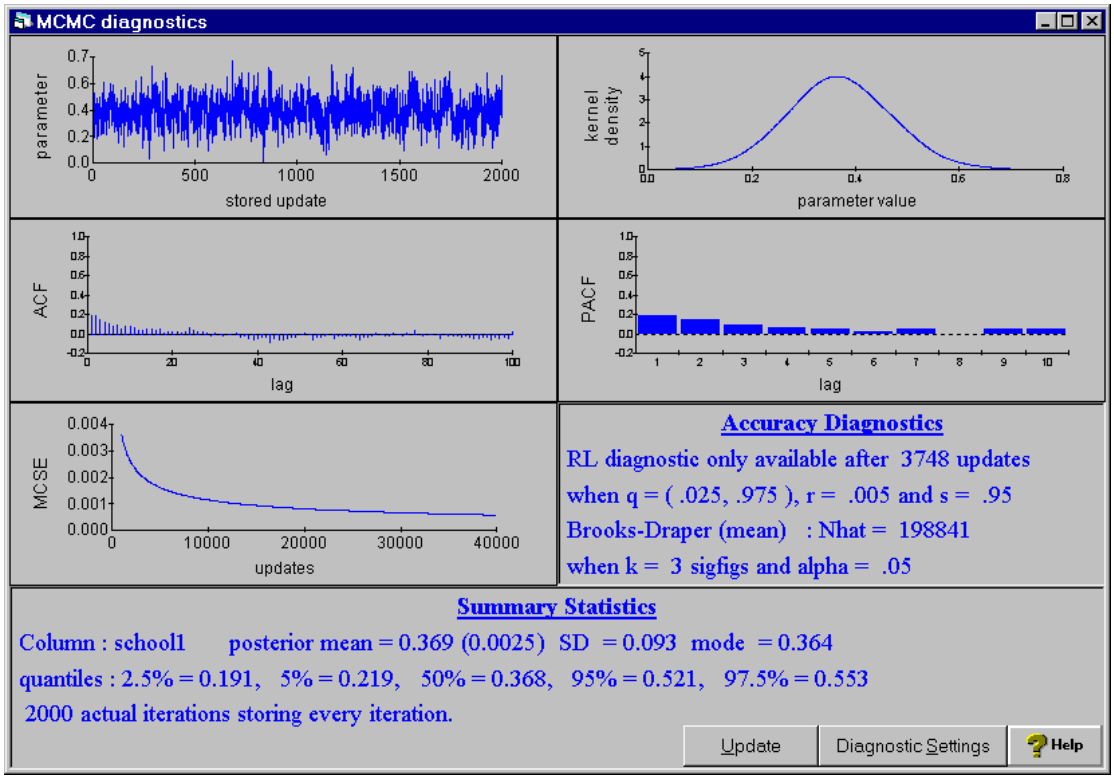
Erase c301-c302

Note that to store 5,000 iterations for each residual would require increasing the worksheet size before running the model

We can name these columns by using the **Names** window and then display the MCMC diagnostics via the **Column diagnostics** window that can be found under the **Basic Statistics** menu.



Choose the column containing the residual for school 1 (c303) that has here been named 'school1' via the Names window and click on **Apply** to see the diagnostics as follows:



As can be seen from the kernel density plot of the residual, this residual has a posterior distribution that is close to Normal, which is what we would expect. This means that we can generate an interval estimate based on the Normal assumption and do not need to use the quantiles.

Note that the Brooks-Draper diagnostic is very large, almost 200,000. This means that if you wanted to be 95% confident of reporting the posterior mean to an accuracy of 3 significant figures (sigfigs), that is how long you would need to run the chain. The MCSE plot to the left shows, in contrast, that if you were willing to report the posterior mean in the form 0.369 ± 0.001, where 0.001 is the Monte Carlo standard error, then only about 15,000 iterations would be needed. (The actual figure is 12,428 = 198,841/16, because to be 95% confident that the posterior mean is 0.369 rather than 0.368 or 0.370, the Monte Carlo interval would have to run from 0.3685 to 0.3695, which implies a Monte Carlo standard error of 0.00025. This is 4 times smaller than 0.001, requiring an increase in run length of $4^2 = 16$).

In further contrast, if you only intent to report the posterior mean to 2 sigfigs (ie 0.37) with 95% Monte Carlo confidence, only 198841/100 ≈ 1989 iterations would be required.

Decreasing the desired value of k (the number of sigfigs) by 1 divides the Brooks-Draper diagnostic by $10^2 = 100$, and increasing k by 1 multiplies it by 100.

Calculating a function of parameters

We have now seen how to calculate accurate interval estimates for residuals. There are however other parameters not estimated directly by the model that could be of interest. In chapter 2, for example, the intra-school correlation, ρ_s was described. This parameter is a function of the level 1 and level 2 variance and so can be calculated from these parameters via the simple formula:

$\rho_s = \sigma^2_u / (\sigma^2_u + \sigma^2_e)$.

Not only can a point estimate be calculated for this parameter but given the chains of the variance parameters, the chain of this function can be constructed and viewed. We firstly run the Gibbs sampler again using the default settings for 5,000 iterations. All the parameters that can be viewed in the **Trajectories** window are stored in a stacked column (in this case c90 is always used) in a similar way to how the residuals were stored in the last section.

In order to calculate the function of parameters we are interested in we will have to firstly unstack column c90. This can be done using the **Generate vector** and **Split column** windows from the **Data manipulation** menu in a similar way to the residuals example in the previous section. Alternatively the **command interface** window can be used and the following commands entered:

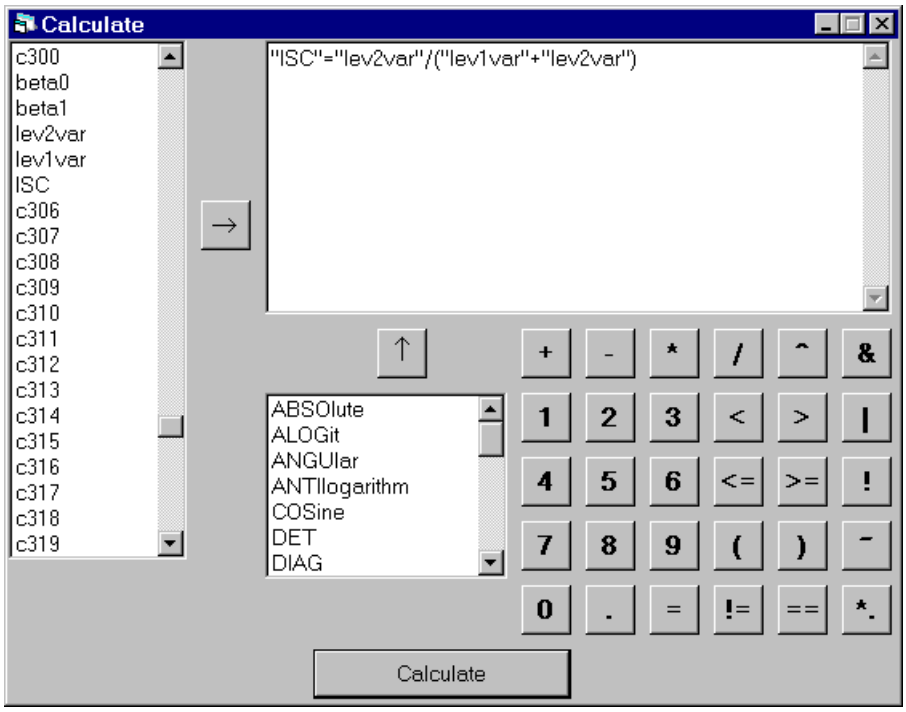
```
code 4 1 5000 c300
split c90 c300 c301-c304
```

Having split the variables into different columns we can then name the columns either by using the **Names** window or again by using the **Command Interface** window by

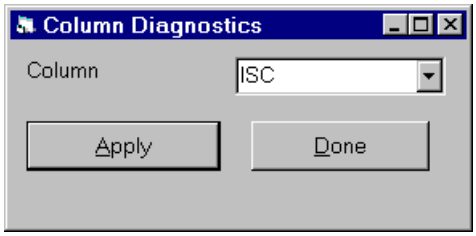
typing the NAME command as follows:

```
name c301 'beta0' c302 'beta1' c303 'lev2var' c304 'lev1var' c305 'ISC'
```

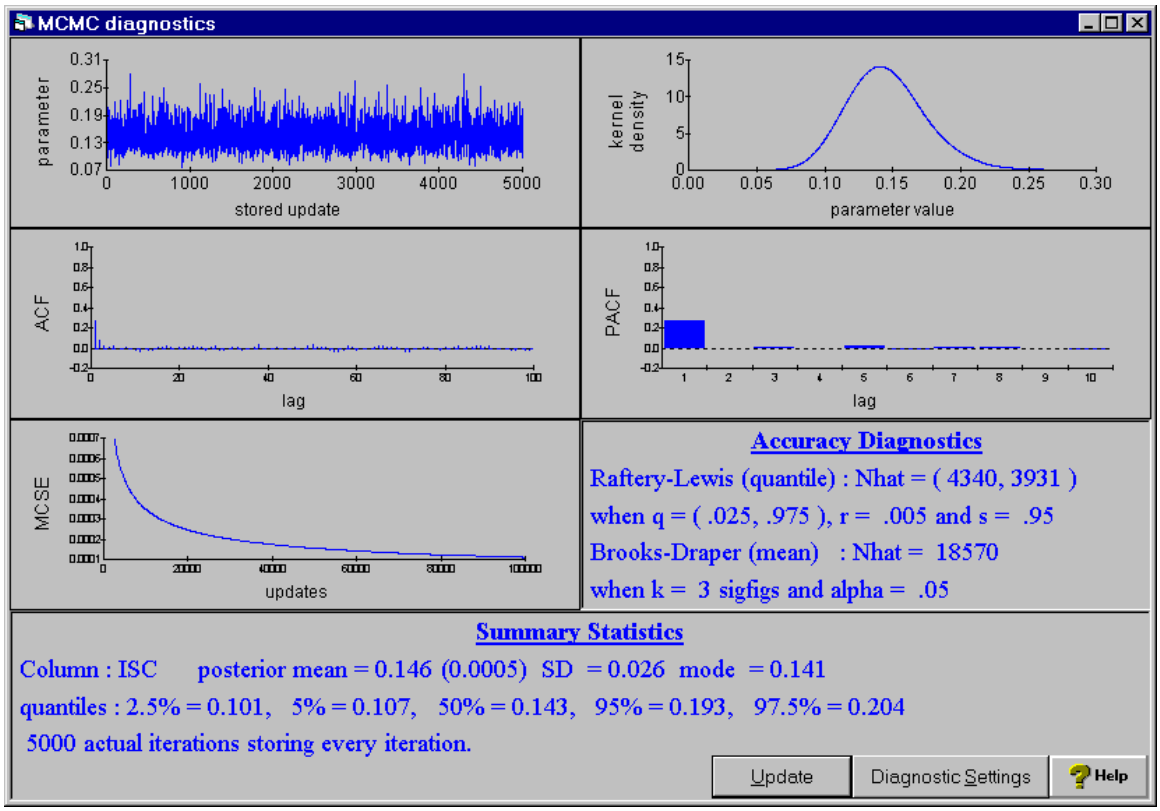
We now need to calculate the chain of values for the intra-school correlation (ISC) and we do this by using the **Calculate** window that can be found in the **Data Manipulation** menu. The column isc should be calculated as follows:



Then after calculating the chain for the intra-school correlation function we now need to use the **Column Diagnostics** window from the **Basic Statistics** menu to display the chain:



Having clicked on **Apply** the diagnostics window should appear as follows:



This window shows that although we are not sampling the derived variable ISC directly we can still monitor its Markov chain. This has the advantage that we can now calculate an accurate interval estimate for this function.

Improving the speed of MCMC Estimation

One feature of MCMC estimation methods is that they are computationally intensive, and take far longer to run than the maximum likelihood IGLS and RIGLS methods. This fact means that any possible speed up of execution will be beneficial. There are two ways to speed up the execution of MCMC estimation methods; firstly to minimise the number of iterations required to give accurate estimates, and secondly to speed up the time for an individual iteration.

One simple procedure to help minimise the number of iterations is to ensure that all continuously distributed explanatory variables are centred at their mean or something close to it. In the example analysed above the reading score predictor has already been standardised to have zero mean. This will minimise correlations among parameters in the posterior, which should increase MCMC accuracy.

The Gibbs sampling method in *MLwiN* has been speeded up in release 1.1. In fact you should find many models will take roughly half the time to run that they took in release 1.0. Although both the **Equations** and **Trajectories** windows are informative to watch while the MCMC methods are running they will both slow down the estimation procedure. For example the following table shows some timings performed on a

Pentium 333MHz PC, running the variance components model for 5,000 iterations after a burn in of 500.

Screen Format	Time
No windows	33 seconds
Equations window	46 seconds
Trajectories window	65 seconds
Both windows	76 seconds

As can be seen displaying the windows, and in particular the **Trajectories** window slows the estimation down.

What you should have learnt from this chapter

How to use the block-updating MH sampler.

How to change starting values for MCMC sampling.

How to choose informative prior distributions and change the default priors.

How to calculate residuals and their chains using MCMC sampling.

How to derive point and interval estimates for functions of model parameters using MCMC sampling

How to speed up MCMC sampling

Chapter 16: Fitting Discrete response models using MCMC

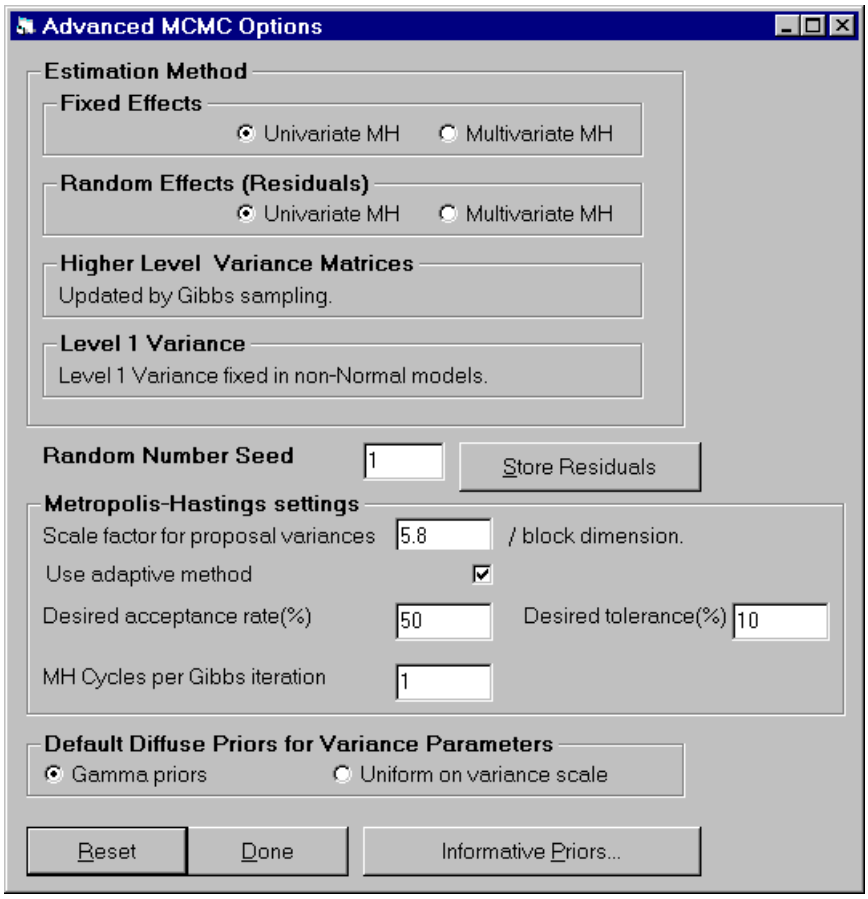
In part II of this manual, we considered how to fit models to datasets where the response variable is not a continuous quantity. We considered two datasets to illustrate two different types of response. We firstly looked at a voting intentions dataset where the response was whether or not a voter voted for the Conservative party in the 1983 general election. The second dataset was taken from a study into the effect of UV exposure on malignant Melanoma mortality. Here the response variable is the count of how many people in each region died.

When we considered fitting discrete response models in the last chapter, we discovered that we could no longer use simple maximum likelihood based techniques, but instead had to use quasi-likelihood techniques. There is also a restriction on the MCMC techniques that can be used on discrete response models but for a different reason. The Normal models discussed in the earlier sections were a special set of models, in that all the parameters in these models have conditional posterior distributions that have standard forms. This means that the standard Gibbs sampling method can be used for all parameters. For discrete response models the conditional posterior distributions for both the fixed effects and the residuals do not have standard forms and consequently MH sampling must be used for these parameters.

In release 1.1 of *MLwiN*, the software knows when the model has a discrete response and if this is the case then only the Metropolis-Hastings (MH) sampling methods can be chosen from the **Advanced MCMC Options** screen.

Voting Example

Consider again the voting dataset (bes83.ws) and set up the simple model with just an intercept term ('Cons') as a fixed effect and random at the area level (For instructions on how to set up this model see Part 2 of the manual.). We then fit this model using RIGLS 1st order MQL estimation to give us starting values as in the previous chapter. Then selecting the MCMC option from the **Estimation control** window will set up the default MCMC settings for a Binomial model. These settings can be seen in the **Advanced MCMC Options** window and should be as follows:



Clicking on **Start** and running the chain for 5,000 monitoring iterations after running the adapting period and a burn in will give the following estimates:

$$\left.\begin{aligned} y_{ij} &\sim \text{Binomial}(n_{ij}, \pi_{ij}) \\ y_{ij} &= \pi_{ij} + e_{0ij}x^{*}_{0} \end{aligned}\right\}$$

$$\text{logit}(\pi_{ij}) = \beta_{1j}x_{1}$$

$$\beta_{1j} = -0.258(0.083) + u_{1j}$$

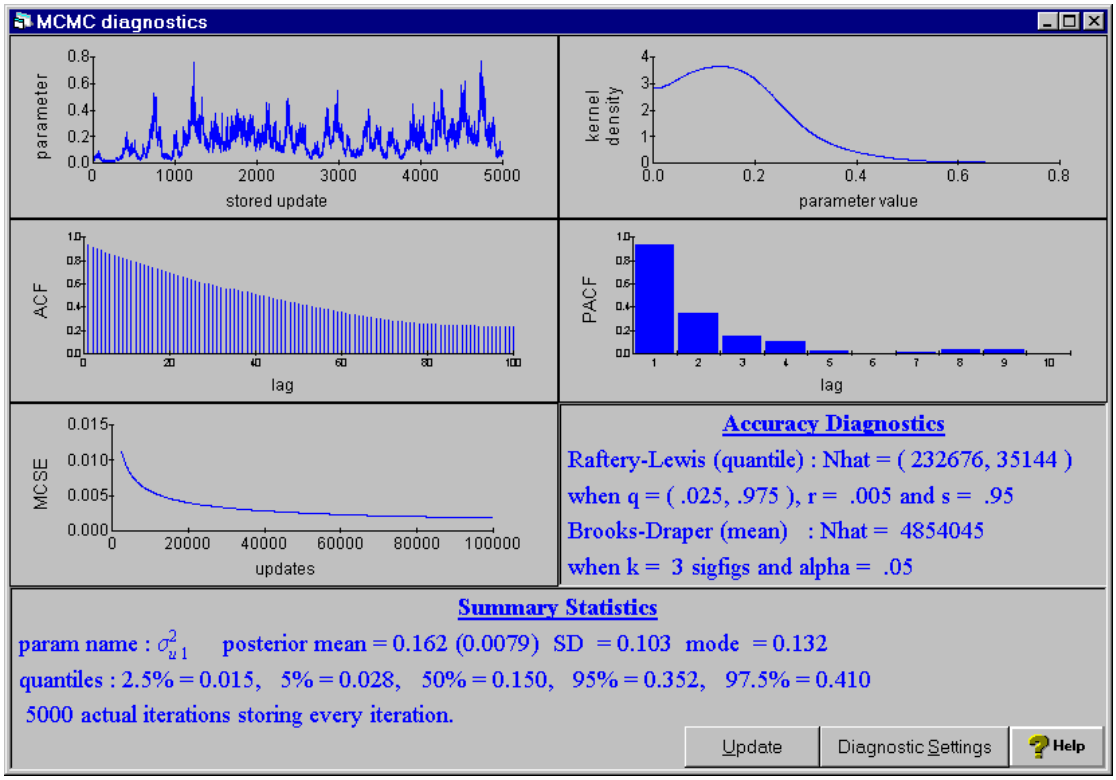
$$\begin{bmatrix} u_{1j} \end{bmatrix} \sim \text{N}(0, \Omega_u) : \Omega_u = \begin{bmatrix} 0.162(0.103) \end{bmatrix}$$

$$x^{*}_{0} = x_{0}[\pi_{ij}(1 - \pi_{ij})/n_{ij}]^{0.5}$$

$$\begin{bmatrix} e_{0ij} \end{bmatrix} \sim (0, \Omega_e) : \Omega_e = \begin{bmatrix} 1.000(0.000) \end{bmatrix}$$

We saw when we fitted the Normal variance components model that the MH sampling method gave higher autocorrelations than the Gibbs sampling method. It is therefore important to check the diagnostics for the parameters here. Click on **Model** and then on **Trajectories** and then click in the displayed trace for the parameter $\sigma^2_{u_1}$. As can be seen in the following diagnostics for the level 2 variance parameter, we have run for only

5,000 iterations but the Raftery-Lewis diagnostic for the 2.5% quantile suggests running for over 200,000 iterations!



The main reason for the huge Nhat value is that the first 200 iterations have the smallest values. The model was now run for 50,000 iterations using both the Gamma prior and the uniform prior and the results can be seen in the following table:

Parameter	MQL 1	PQL 2	MCMC (Gamma)	MCMC (Uniform)
β_1	-0.248 (0.081)	-0.257 (0.082)	-0.256 (0.082)	-0.262 (0.088)
$\sigma^2_{u_1}$	0.134 (0.091)	0.150 (0.103)	0.137 (0.108)	0.218 (0.114)

The Raftery-Lewis Nhat statistic was calculated for both parameters using both methods and it was found that the statistic was always smaller than 50,000. What is interesting is that for this model the Gamma prior gives a smaller variance estimate than the PQL method. This behaviour has been observed (Browne 1998) for Normal models when the level 2 variance is much smaller than the level 1 variance. Unfortunately, for binary models it is difficult to compare the level 1 and level 2 variance parameters. One crude way of doing this would be to forget that the data are binary and fit the model as a Normal response model.

Fitting this model as a Normal model using IGLS we get the following estimates:

$$y_{ij} \sim \mathrm{N}(XB, \Omega)$$

$$y_{ij} = \beta_{0ij} x_0$$

$$\beta_{0ij} = 0.438(0.020) + u_{0j} + e_{0ij}$$

$$\begin{bmatrix} u_{0j} \end{bmatrix} \sim \mathrm{N}(0, \Omega_u) : \Omega_u = \begin{bmatrix} 0.009(0.006) \end{bmatrix}$$

$$\begin{bmatrix} e_{0ij} \end{bmatrix} \sim \mathrm{N}(0, \Omega_e) : \Omega_e = \begin{bmatrix} 0.237(0.013) \end{bmatrix}$$

*-2*log(like)* = 1145.724(800 of 800 cases in use)

Here we clearly see that the level 2 variance is only a small proportion (3.7%) of the total variance and is also of very small magnitude. This casts some doubt on the validity of fitting this model as a multilevel model and explains the difficulties that the MCMC methods have in estimating the level 2 variance parameter and hence the differences between the two priors. Fitting the data as a Normal model also gives the estimated proportion of voters voting Conservative (43.8%) directly.

If we now fit the Binomial model with the 4 additional predictors (see Part 2 of the manual for instructions on setting up this model) using both MCMC priors and monitoring runs of 50,000 iterations we will get results similar to the following table:

Parameter	MQL 1	PQL 2	MCMC (Gamma)	MCMC (Uniform)
β_1	-0.355 (0.092)	-0.367 (0.095)	-0.365 (0.091)	-0.377 (0.102)
β_2	0.089 (0.018)	0.093 (0.018)	0.091 (0.018)	0.095 (0.019)
β_3	0.067 (0.013)	0.069 (0.014)	0.069 (0.014)	0.070 (0.014)
β_4	0.044 (0.019)	0.046 (0.019)	0.046 (0.019)	0.046 (0.020)
β_5	0.138 (0.018)	0.143 (0.018)	0.142 (0.018)	0.146 (0.019)
$\sigma^2_{u_1}$	0.144 (0.114)	0.167 (0.119)	0.096 (0.113)	0.261 (0.152)

Here again the level 2 variance estimates from MCMC are different from MQL and PQL because the level 2 variance is so small, but the fixed effects estimates are similar for all four methods. Fitting this model as a Normal response model shows the level 2 variance to be an even smaller proportion (2.6%) of the total variation again casting

doubt on the use of multilevel modelling on this dataset.

As demonstrated at the end of the Chapter 8 the model that only includes an intercept term can also be fitted as a general Binomial model with 110 proportions as responses. When this is done using MCMC using a monitoring run of 50,000 iterations we get essentially the same estimates. This was also seen for the RIGLS 2nd order PQL method:

Parameter	MCMC (gamma prior)	MCMC (uniform prior)
β_0	-0.256 (0.082)	-0.262 (0.088)
σ^2_u	0.137 (0.108)	0.218 (0.114)

The main thing to notice when fitting this model as a proportion is the significant speed increase of the MCMC method.

Melanoma mortality dataset

The second dataset discussed in Chapter 9 was a Poisson example in which the response was the number of malignant Melanoma deaths in 9 European community countries (mmmec.ws). In release 1.1 of *MLwiN*, Poisson response models can also be fitted using the Metropolis-Hastings MCMC sampling method.

We will consider here the three level model containing 'uvbi' as a fixed effect (see Part 2 of the manual for instructions on how to set up this model). The RIGLS 2nd order PQL estimates were used as starting values for the MH method. The Poisson dataset is fairly small and so estimation is quick. The estimates achieved after running for 50,000 iterations using the adaptive method and the Gamma prior are as follows:

$$\left.\begin{aligned} y_{ijk} &\sim \text{Poisson}(\pi_{ijk}) \\ y_{ijk} &= \pi_{ijk} + e_{1ijk}x^{*}_{1} \end{aligned}\right\}$$

$$\log(\pi_{ijk}) = \beta_{0jk}x_{0} + \text{-0.027(0.011)}x_{2ijk}$$

$$\beta_{0jk} = \text{-0.067(0.146)} + v_{0k} + u_{0jk}$$

$$\begin{bmatrix} v_{0k} \end{bmatrix} \sim \text{N}(0,\ \Omega_{v}) \ :\ \Omega_{v} = \begin{bmatrix} \text{0.203(0.141)} \end{bmatrix}$$

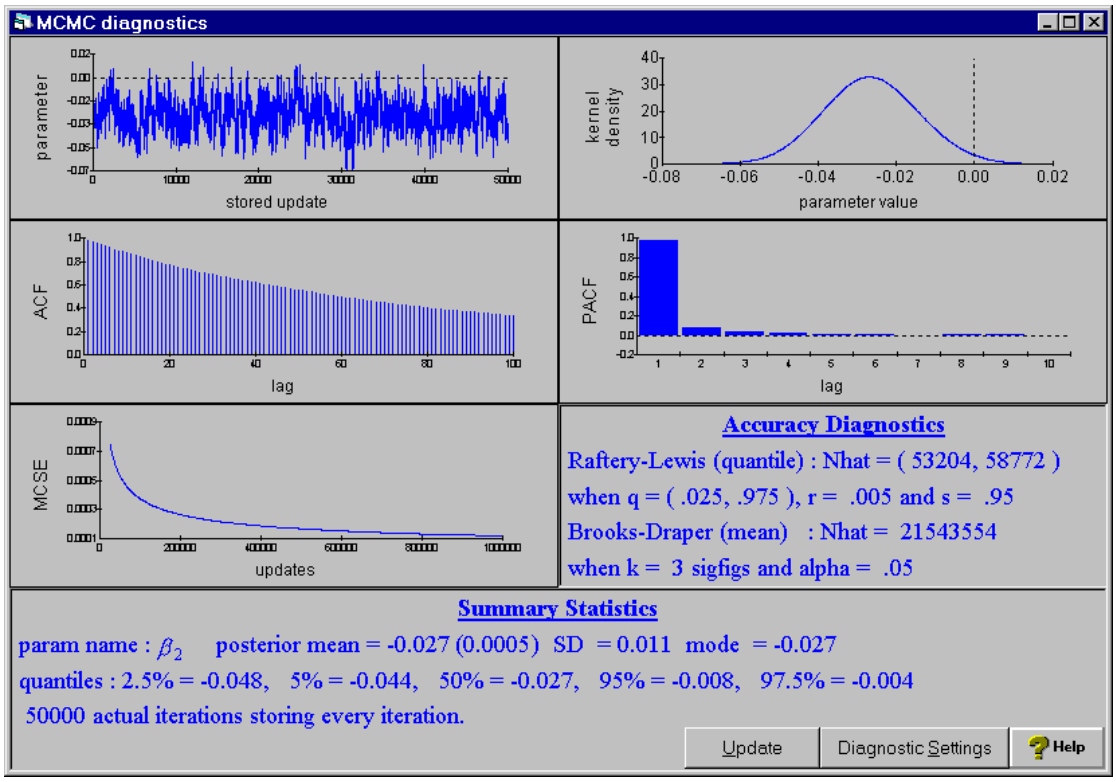
$$\begin{bmatrix} u_{0jk} \end{bmatrix} \sim \text{N}(0,\ \Omega_{u}) \ :\ \Omega_{u} = \begin{bmatrix} \text{0.051(0.012)} \end{bmatrix}$$

$$x^{*}_{1} = x_{1}\pi_{ijk}^{0.5}$$

$$\begin{bmatrix} e_{1ijk} \end{bmatrix} \sim (0,\ \Omega_{e}) \ :\ \Omega_{e} = \begin{bmatrix} \text{1.000(0.000)} \end{bmatrix}$$

These estimates are similar to the results seen for the 2nd order PQL estimation procedure. The level 3 (nations) variance is quite a bit larger but this is mainly due to there being only 9 level 3 units making this parameter's posterior distribution heavily skewed.

Note that the fixed effects in this Poisson example appear to give large Raftery Lewis Nhat values, for example parameter β_2:



All the other models fitted in Chapter 9 to the Melanoma dataset can also be fitted using MCMC but we will not discuss them here.

What you should have learnt from this chapter

That both Binomial and Poisson models can be fitted using MCMC sampling

That the MH sampler has to be used for the fixed effects and the residuals when fitting discrete response models using MCMC sampling in *MLwiN*.

Chapter 17: Bootstrap Estimation

In the last 3 chapters we have studied the first of the two simulation based families of methods available in *MLwiN*, MCMC sampling and we will now consider the other one. We have already introduced the idea of bootstrapping for a simple one level problem. In multilevel modelling the bootstrap can be used for two main purposes. First, as an alternative procedure to MCMC methods, to make accurate inferences on the basis of simulated parameter estimates. Thus, for example, while in Normal response models we can construct confidence intervals for functions of the fixed parameters assuming Normality, this may not be appropriate for the random parameters unless the number of units at the level to which the parameter refers is large. For an introduction to bootstrapping see Efron and Tibshirani (1993). In version 1.1 of *MLwiN*, both a parametric bootstrap procedure and a new non-parametric bootstrap procedure can be used.

In the simulation introductory chapter we saw how bootstrapping was used to construct several simulated datasets from the original dataset and/or model parameters. Then estimates of the parameters of interest were found for each of these new datasets and hence a chain of values was obtained that allowed distributional summaries to be found. In multilevel modelling the procedure is similar. The bootstrapping methods are used to construct the bootstrap datasets and then either the IGLS or RIGLS estimation method can be used to find estimates for each dataset. The parametric bootstrap works exactly as in chapter 1 in that the datasets are generated (by simulation) based on the parameter estimates for the original dataset. Due to the multilevel structure of the problems fitted in *MLwiN* we cannot use the simple non-parametric approach introduced in chapter 1, but instead we will introduce a new approach based on sampling from the estimated residuals.

The second purpose for which bootstrap estimation can be used is to correct any bias in the parameter estimation (again as an alternative to MCMC methods). This is useful in models with discrete responses where the standard estimation procedure based upon quasilikelihood estimation produces estimates, especially of the random parameters, that are downwardly biased when the corresponding number of units is small (see Goldstein and Rasbash, 1996 for a discussion). The severity of this bias can be trivial in some data sets and severe in other data sets. A complicating feature in these models is that the bias is a function of the underlying 'true' value so that the bias correction needs to be iterative. In the next section we illustrate how this works.

Understanding the iterated bootstrap

Suppose we simulate a data set for a simple variance components model where the standard *MLwiN* estimation procedure has a downward bias of 20% for the level 2 variance parameter, σ_u^2. Suppose also that the true value for the level 2 variance is 1.0.

Then if we estimate this model for several simulated datasets using the standard procedure we will obtain an average estimate of 0.8.

Imagine now that we have just one simulated data set with a level 2 variance estimate that happens to be 0.8, together with fixed parameter estimates to which we can apply the same procedure. We can now simulate (parametrically bootstrap) a large number of new response vectors from the model with level 2 variances of 0.8 and calculate the average of the variance estimates across these new replicates. We would expect a value of 0.64 since the level 2 variance is estimated with a downward bias of 20% (0.8*0.8 =0.64). Now if we add the downward bias of 0.16 to our starting value of 0.8 we obtain a *bias corrected* estimate 0.96. We can now run another set of simulations this time taking the bias corrected estimates (0.96 for the variance) as our starting simulation values. Averaging across replicates we now expect an average of 0.768 for the variance parameter which results in a bias estimate of 0.192. We then add this estimated bias to 0.8 to give a bias corrected estimate of 0.992. We can now go on to simulate another set of replicates from the latest bias corrected estimate and repeat until the successive corrected estimates converge (see table below). We shall see how we can judge convergence in the example that follows. Note that in models where the bias is independent of the underlying true value (additive bias) only a single set of bootstrap replicates is needed for bias correction.

Replicate Set	Starting Value	Simulated Estimate (Standard procedure)	Estimate (Bias corrected)
1	0.8	0.8*0.8 = 0.64	0.8+(0.8-0.64) = 0.96
2	0.96	0.96*0.8 = 0.768	0.8+(0.96-0.768) = 0.992
3	0.992	0.992*0.8 = 0.7936	0.8+(0.992-0.7936) = 0.9984
4	0.9984	0.9984*0.8 = 0.7987	0.8+(0.9984-0.7987) = 0.9997
5	0.9997	0.9997*0.8 = 0.7997	0.8+(0.9997-0.7997) = 1.0000

With this release of *MLwiN* there is still relatively little experience of using bootstrap methods with multilevel models. We suggest therefore that this procedure should be used with care. Bootstrap estimation is based on simulation and therefore convergence is stochastic. This raises the question of what is a large enough number of replicates in each bootstrap *set*. On the examples tried, replicate sets of between 300-1000 replicates and a series of about 5 sets is usually sufficient to achieve convergence. This adds up to a substantial amount of computation. For this reason bootstrapping, like MCMC estimation should not be used for model exploration, but rather to obtain unbiased estimates and more accurate interval estimates at the final stages of analysis.

At convergence the current replicate set can be used to generate confidence intervals or any other desired descriptive statistic for model parameters (see below)

An example of bootstrapping using *MLwiN*

Following on from chapter 8 we will work with the bes83.ws data set we originally used in the chapter on binary response models. Retrieve this data set and set up the 2 level variance components model, with 'voter' as the level 1 identifier, 'area' as the level 2 identifier and 'cons' 'defence', 'unemp', 'taxes' and 'privat' as explanatory variables. You will need to use the variable 'bcons' to model the binomial variation at level 1, as described in the chapter on binary response models. If you run this model as a first order, MQL, RIGLS model, you will obtain the following results:

$$\left.\begin{aligned} y_{ij} &\sim \text{Binomial}(n_{ij}, \pi_{ij}) \\ y_{ij} &= \pi_{ij} + e_{0ij}x^{*}_{0} \end{aligned}\right\}$$

$$\text{logit}(\pi_{ij}) = \beta_{1j}x_{1} + 0.089(0.018)x_{2ij} + 0.067(0.013)x_{3ij} + 0.044(0.019)x_{4ij} + 0.138(0.018)x_{5ij}$$

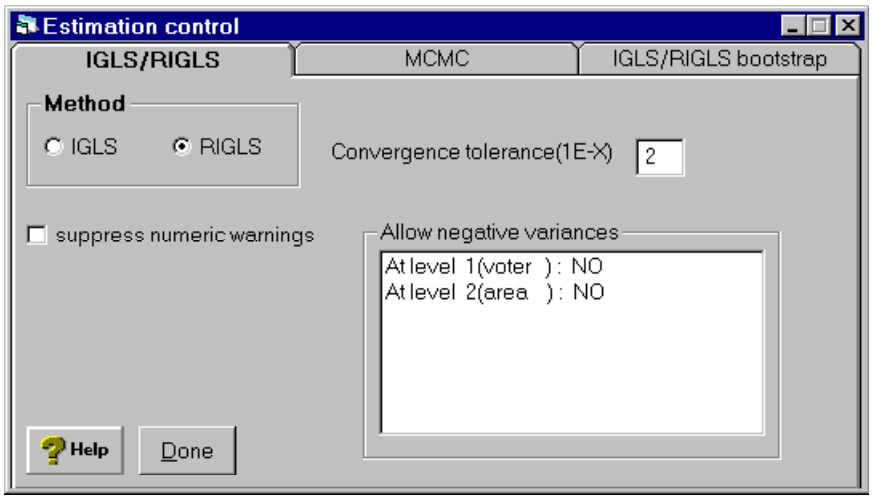
$$\beta_{1j} = -0.355(0.092) + u_{1j}$$

$$\begin{bmatrix} u_{1j} \end{bmatrix} \sim \text{N}(0, \Omega_u) : \Omega_u = \begin{bmatrix} 0.144(0.114) \end{bmatrix}$$

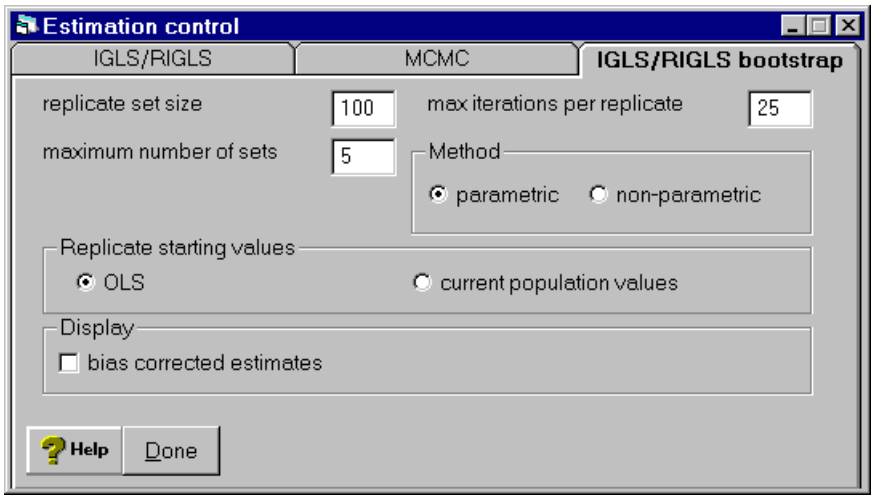
$$x^{*}_{0} = x_{0}[\pi_{ij}(1 - \pi_{ij})/n_{ij}]^{0.5}$$

$$\begin{bmatrix} e_{0ij} \end{bmatrix} \sim (0, \Omega_e) : \Omega_e = \begin{bmatrix} 1.0(0.0) \end{bmatrix}$$

You may wish to experiment with a range of bootstrapping options using this as the base model, so save this model in a worksheet so you can return to it at a later stage. To set up a bootstrap run click on the **Estimation control** button on the main toolbar and the following window will appear:



In the **Allow negative variances** box click on both level 1 and level 2 to allow negative variances at both levels. This retains any negative variances that occur in individual bootstrap replicates, rather than setting them to zero, so that a consistent bias correction is estimated. We cam now click on the **IGLS/RIGLS** bootstrap tab which brings up the following screen:



We will initially use the parametric bootstrap and talk about the non-parametric bootstrap in a later section. Here we can also set the number of replicates per set. We can also set the maximum number of iterations per replicate. If a replicate has not converged after the specified number of iterations (these are standard *MLwiN* iterations) the replicate is discarded. *MLwiN* does not judge a set of replicates to be completed until the full quota of converged replicates has been run. While the bootstrap is running a progress count is maintained on the bottom of the screen and the number of discarded replicates is also reported. If this count grows large you may wish to restart the bootstrap with a higher setting for maximum number of iterations per replicate. We will initially use the displayed default settings, so now click on the **Done** button.

We want to watch the progress of the bootstrap as estimation proceeds so now choose the **Trajectories** window from the **Model** menu. The parameter that exhibits the most bias is the level 2 (between area) variance. We will set the trajectories window to show the graph for this parameter only. This is achieved by pressing the **Select** button and choosing the **'area:cons/cons'** item from the list that appears. Also choose '**1 graph per row**' from the pull down list at the bottom of the window and then click on the **Done** button. This one graph should now fill the entire Trajectories window. Note that opening the Trajectories window will slow down the iterations.

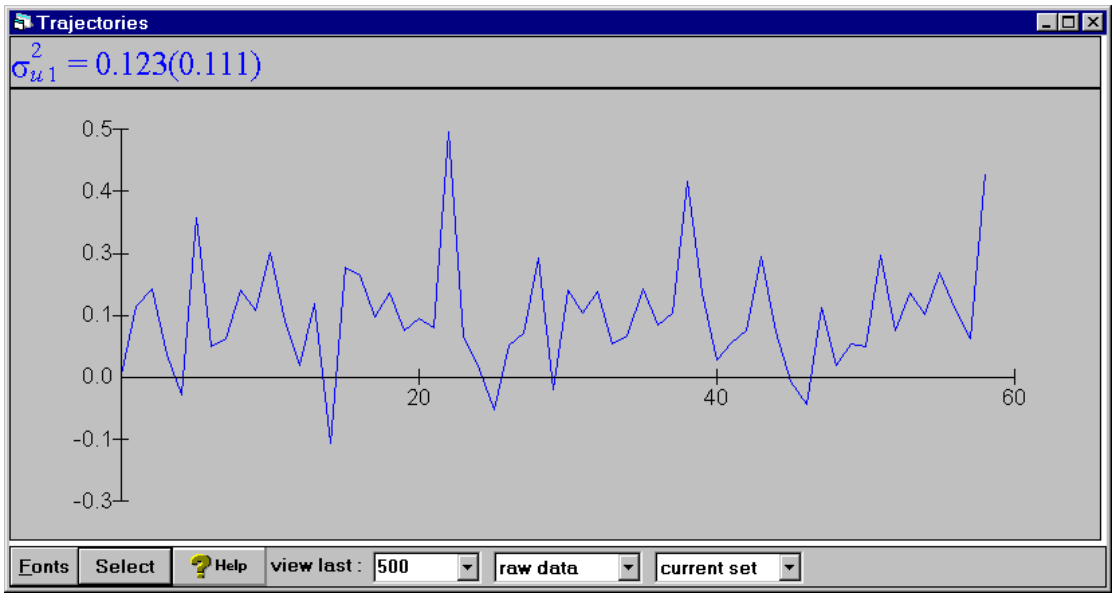
All the bootstraps shown in this chapter were run with a seed for the random number generator of 100. If you wish to produce exactly the same results, you can set the random number seed by opening the **Command interface** window and typing the command

seed 100

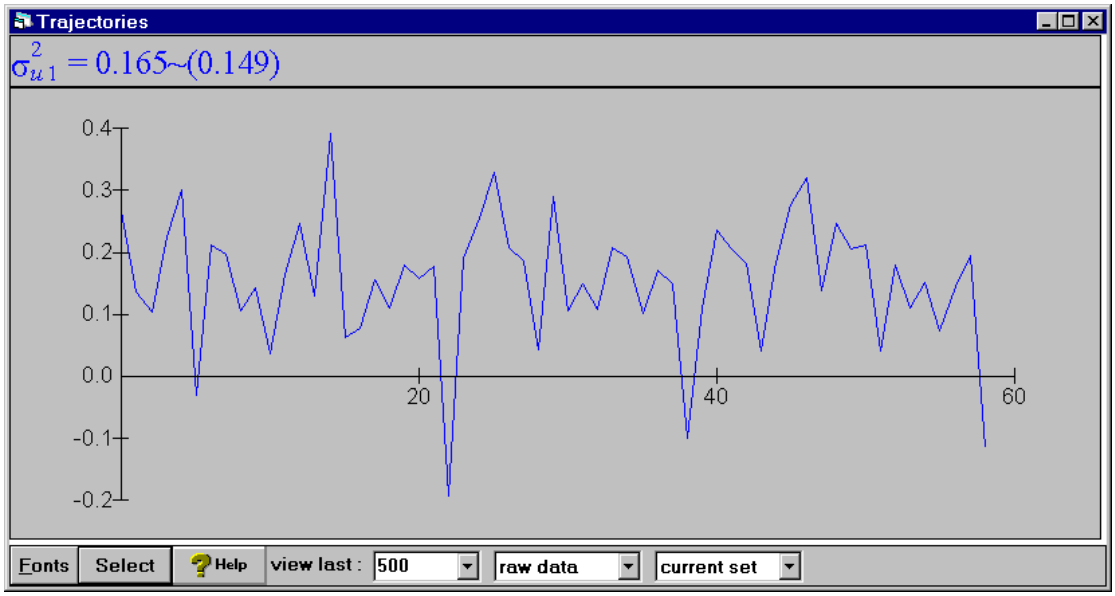
You may prefer to set a different seed or to let *MLwiN* choose its own seed value.

Note that it is important that you do not change any other settings for the model after running to convergence using quasilikelihood, such as switching from PQL to MQL etc.

We can now set the bootstrap running by pressing the **Start** button on the main toolbar. The trajectories window will display the bootstrap chain for the current replicate set. After approximately 60 replicates the bootstrap chain for the first replicate set will look somewhat like this:



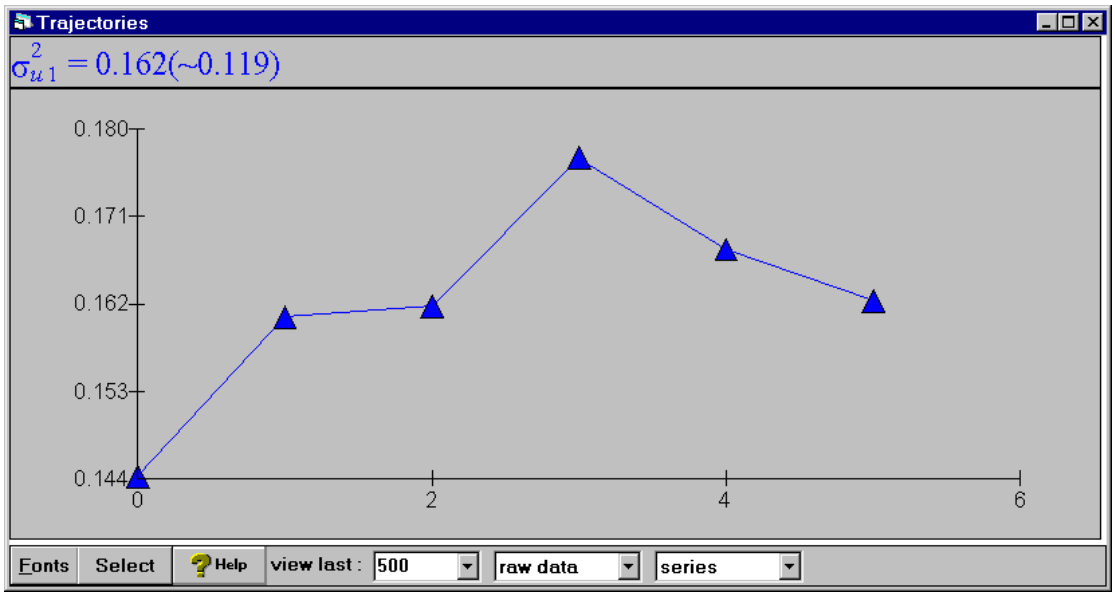
Note that since we are allowing negative variances some of the points cross the x-axis. By default the values shown are not corrected for bias. We can at any time switch between bias corrected and uncorrected estimates by opening the **Estimation control** window and checking the **bias corrected** box. When the bias correction box is checked another checkbox labelled **scaled SE's** appears. If you check both these boxes you will observe that the **Trajectories** window changes as follows:



The current bootstrap estimate moves up, in this case, from 0.123 to 0.165. Note that we started with the RIGLS estimate of 0.144 and hence the bias corrected estimate is simply calculated as 0.144 + (0.144-0.123) = 0.165. The scaled SE's option only changes the reported standard error shown in brackets on the Trajectories title bar. This scaling process ensures that standard errors and quantile estimates for bias corrected estimates are properly scaled. The scaling is an approximation and hence scaled standard errors and quantiles are preceded by a tilde (~). See the **Help** system for more details.

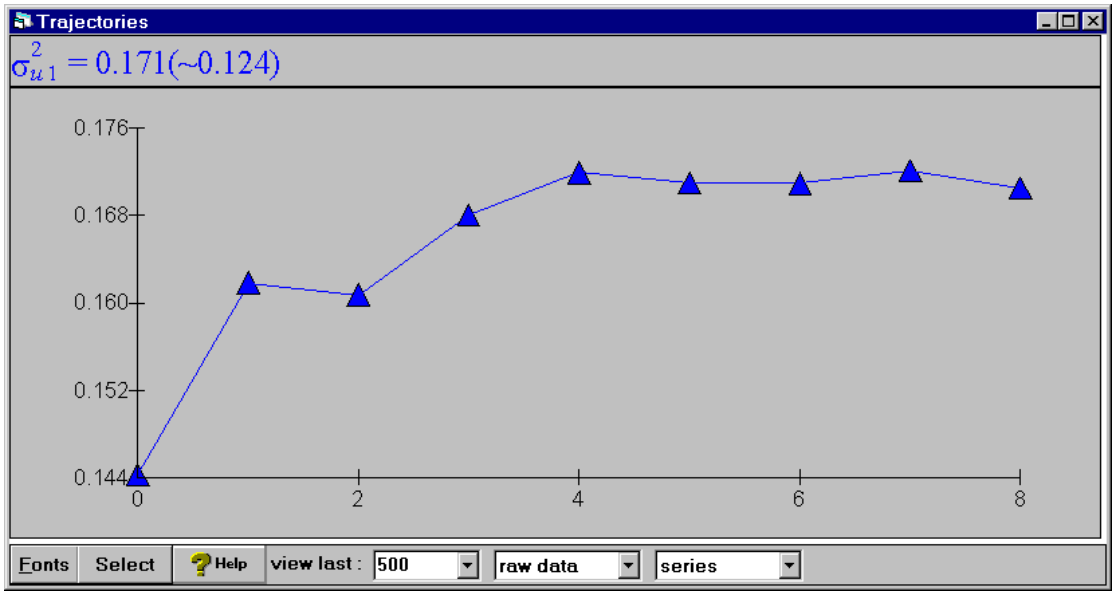
It is useful to view bootstrap replicate sets as a sequence of running means because this gives some clues as to convergence. You can do this by clicking on the drop down list box on the trajectories window that has "**raw data**" currently selected. This allows you to switch between viewing the current bootstrap replicate chain as raw data or as a running mean. A converged running mean chain should be reasonably stable.

At any point you can move from viewing the current replicate set chain to the series of replicate set summaries by selecting accordingly from the drop down list on the **Trajectories** window tool bar that has the options **current set** and **series**. When viewing the series of set summaries it is more informative to select **raw data** rather than **running mean**. After five sets of bootstrap replicates the series graph looks like this:



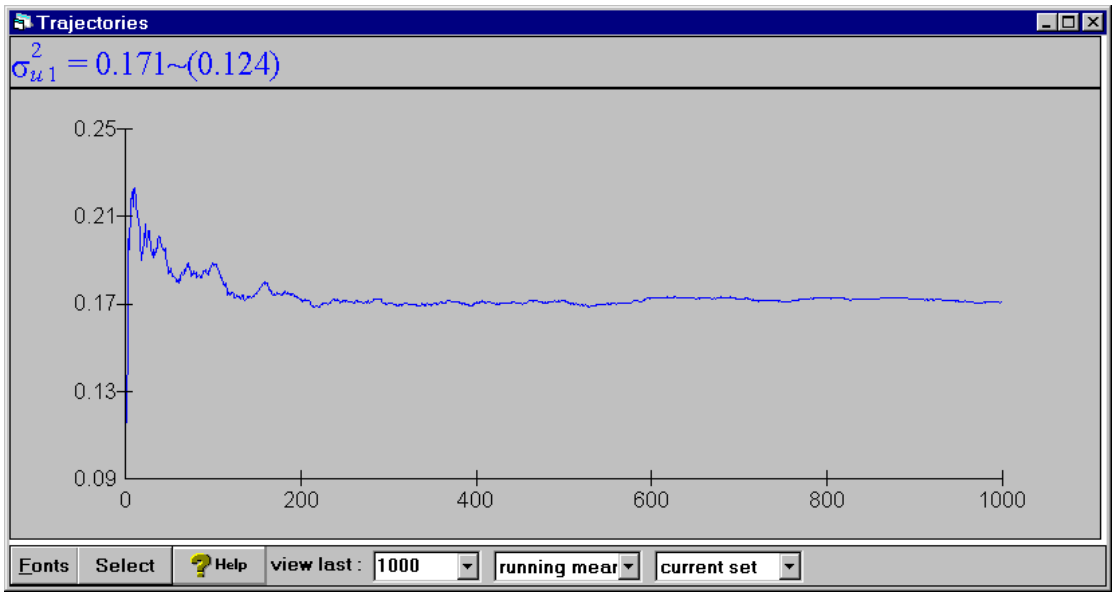
This graph is useful for judging bootstrap convergence. If we are viewing bias corrected estimates we would expect to see this graph levelling out if the bootstrap series has converged. This is not the case here. This means we probably need to increase the replicate set size to reduce simulation noise.

Below is the equivalent graph for a bootstrap on this data set where the replicate set size has been increased from 100 to 1000 and the number of replicate sets from 5 to 8. Note that for this dataset and model, running the bootstrap for this long will take several hours on most machines.



As we might have expected this graph is less erratic and we can therefore have more confidence that the bootstrap has converged. In fact it looks reasonably stable after set 4. From an original estimate of 0.144 the bootstrap eventually produces a bias corrected estimate of 0.171.

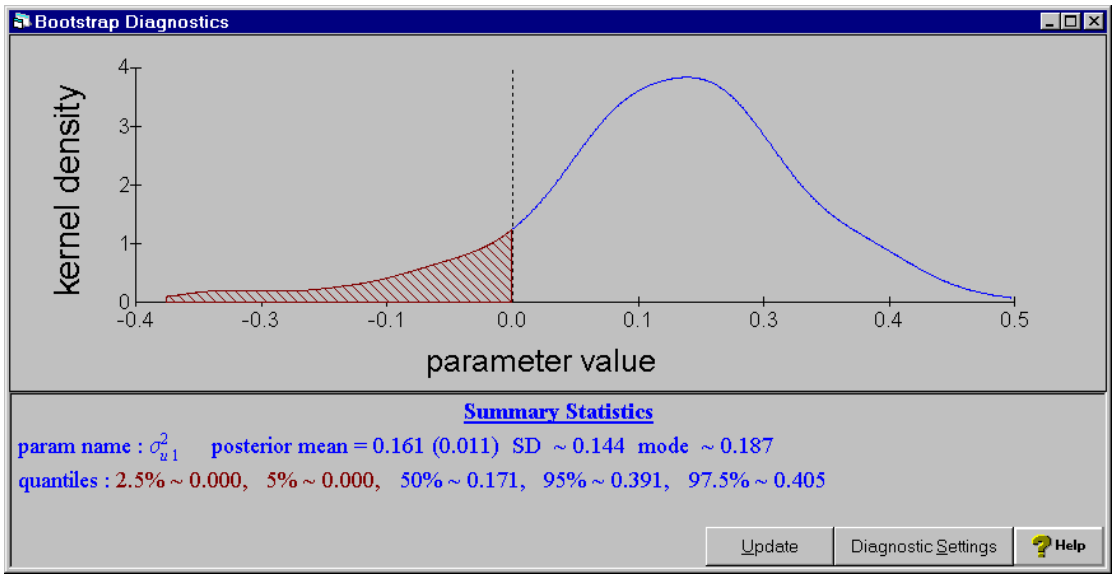
The running mean for the last replicate set on this bootstrap is:



The **running mean** is selected from the middle pull down option box and the **view last** box is increased to 1000 to see the whole chain. We see that this replicate set stabilised between 200-400 replicates. This means that on this example we should have been able to use a replicate set size of 400 and a series of 5 sets. It is generally sensible, however, to be over cautious in selecting replicate set sizes and series lengths.

Diagnostics and confidence intervals

At any stage in the bootstrap, when viewing a replicate set chain we can obtain a kernel density plot and quantile estimates that are calculated from the chain. This is achieved by clicking on the graph in the **Trajectories** window for the parameter we are interested in (in a similar way to MCMC sampling, see the **help** system for further details). Below are the diagnostics obtained from a (bias corrected) replicate set of the bootstrap with 100 replicates per set:



Firstly notice the irregularity of the kernel density plot which is due to having too few replicates per set - in this case 500 may be a more suitable number. The second thing to notice is that the area below zero on the x-axis is shaded (in red). This is because we are viewing a kernel density for a variance parameter. While we allow negative estimates for variances during bootstrap estimation to ensure consistent bias correction, when we come to calculate quantiles and summary statistics we set to zero any results that are negative.

Non parametric bootstrapping

When we considered the single level example in Chapter 13 with a sample of 100 heights, it was easy to perform a non-parametric bootstrap. We simply drew samples of size 100 with replacement from the 100 heights. When we consider a multilevel model where the responses come from different higher level units an analogous approach is problematic.

Consider the tutorial example covered in the first section of the manual, where we had 4059 students in 65 schools. If we were simply to sample pupils with replacement then we would generate datasets that do not have the same structure as our original dataset, for example school 1 may have 10 pupils but in the first simulated dataset 15 pupils or even worse no pupils could be generated. If on the other hand we were to sample with replacement within each school then this is also problematic as some schools have very few pupils to sample from.

The approach introduced in version 1.1 of *MLwiN* involves resampling from the estimated residuals (as opposed to the responses) in the model. It may more precisely be referred to as a semi-parametric bootstrap since the fixed parameters are used. The

mathematical procedure used involves sampling from the 'unshrunken' residuals to produce the correct variance estimates. The procedure is included here for completeness and is also described in the help system.

The resampling procedure

Consider the 2 level model

$$y_{ij} = (X\beta)_{ij} + (ZU)_j + e_{ij}$$
$$U^T = \{U_0, U_1 \ldots.\}$$

Having fitted the model we estimate the residuals at each level as

$$\hat{U} = \{\hat{u}_0, \hat{u}_1 \ldots.\}, \quad \hat{e}$$

If we were to use these residuals directly to sample from, we will underestimate the variance parameters because of the shrinkage. It is the case that the correlation structure among the estimates within and between levels reproduces the correct total variance when estimated residuals are used. However the random sampling with replacement upon which bootstrap sampling of the residuals is based will not preserve this structure and so will not generally produce unbiased estimates of either the individual random parameters or of the total variance and covariance of responses.

One possibility, shown already in this chapter is to use a fully parametric bootstrap. This, however, has the disadvantage of relying upon the Normality assumption for the residuals. We use instead a resampling estimated residuals procedure which produces unbiased distribution function estimators, as follows.

For convenience we shall illustrate the procedure using the level 2 residuals, but analogous operations can be carried out at all levels. Write the empirical covariance matrix of the estimated residuals at level 2 in model (1) as

$$S = \frac{\hat{U}^T\hat{U}}{M}$$

and the corresponding model estimated covariance matrix of the random coefficients at level 2 as *R*. The empirical covariance matrix is estimated using the number of level 2 units, *M,* as divisor rather than *M-1*. We assume that the estimated residuals have been centered, although centering will only affect the overall intercept value.

We now seek a transformation of the residuals of the form

$$\hat{U}^* = \hat{U}A$$

where A is an upper triangular matrix of order equal to the number of random coefficients at level 2, and such that

$$\hat{U}^{*T}\hat{U}^{*} = A\hat{U}^{T}\hat{U}A = A^{T}SA \ = R$$

The new set of transformed residuals $\hat{U}^{*}$ now have covariance matrix equal to that estimated from the model, and we sample sets of residuals with replacement from $\hat{U}^{*}$. This is done at every level of the model, with sampling being independent across levels.

To form A we note that we can write the Cholesky decomposition of S, in terms of a lower triangular matrix as $S = L_S L_S^T$ and the Cholesky decomposition of R as $R = L_R L_R^T$.

We then have $L_R L_S^{-1}\hat{U}^{T}\hat{U}(L_R L_S^{-1})^{T} = L_R L_S^{-1} S (L_S^{-1})^{T}(L_R)^{T} = L_R (L_R)^{T} = R$

Thus, the required matrix is

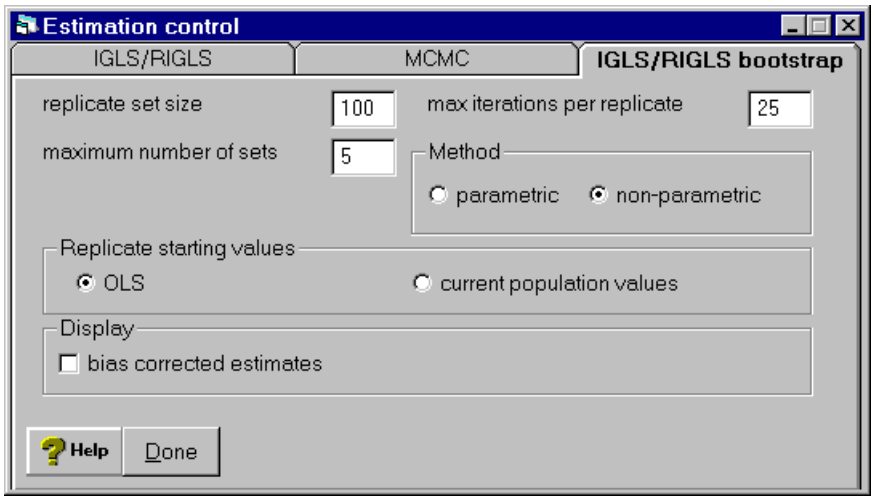
$$A = (L_R L_S^{-1})^{T}$$

and we can hence find the $\hat{U}^* = \hat{U}A$ and then use then to bootstrap non-parametrically a new set of level 2 residuals. *MLwiN* automatically carries out these calculations when using the non-parametric procedure.

The Voting dataset example

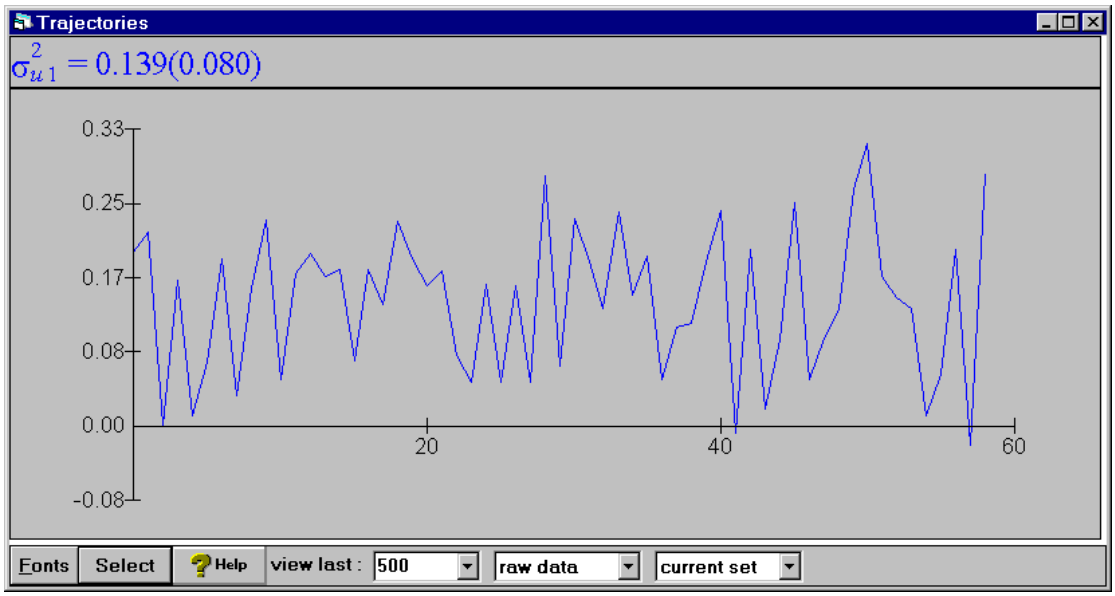
Although the non-parametric bootstrap has a different method for creating the bootstrap datasets, the two methods both produce chains of parameter values. We will now repeat our analysis of the voting intention dataset (bes83.ws) using the non-parametric bootstrap. The random number seed 100 was again used to generate the results shown in this section.

To start the example, retrieve the bes83.ws worksheet, set up the model and run the first order MQL RIGLS model as in section 5.3. Note that if you have just been following the parametric bootstrap example the model is already set up and you simply need to change estimation method to RIGLS and rerun the model. We now want to select the bootstrap method so click on the **Estimation control** button on the main toolbar and the IGLS/RIGLS options will appear. Ensure that the **Allow negative variances** box has **Yes** set at both level 1 and 2 before clicking on the IGLS/RIGLS bootstrap tab. We now want to select the non-parametric bootstrap option from the **Method** box. We will leave the other parameters at their default values so that the window appears as below. Having set the parameters click on **Done** to continue.

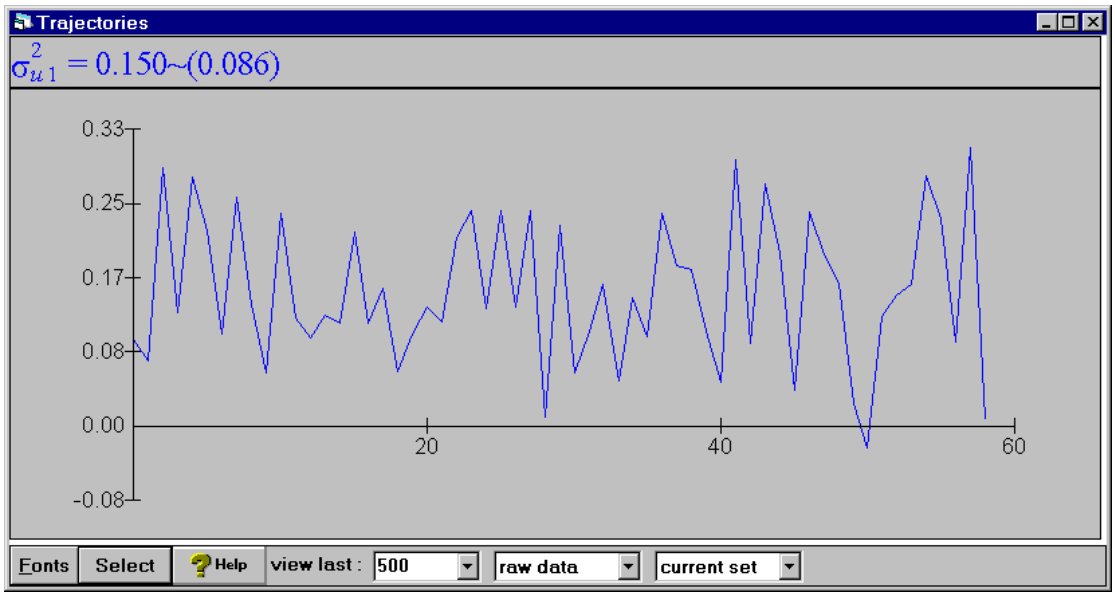


We can now repeat the analysis performed in section 5.3 with the non-parametric bootstrap.

Click on the **Start** button on the main toolbar to set the non-parametric bootstrap running. After 60 or so replicates the bootstrap chain for the first replicate set of the level 2 variance parameter should look like this:

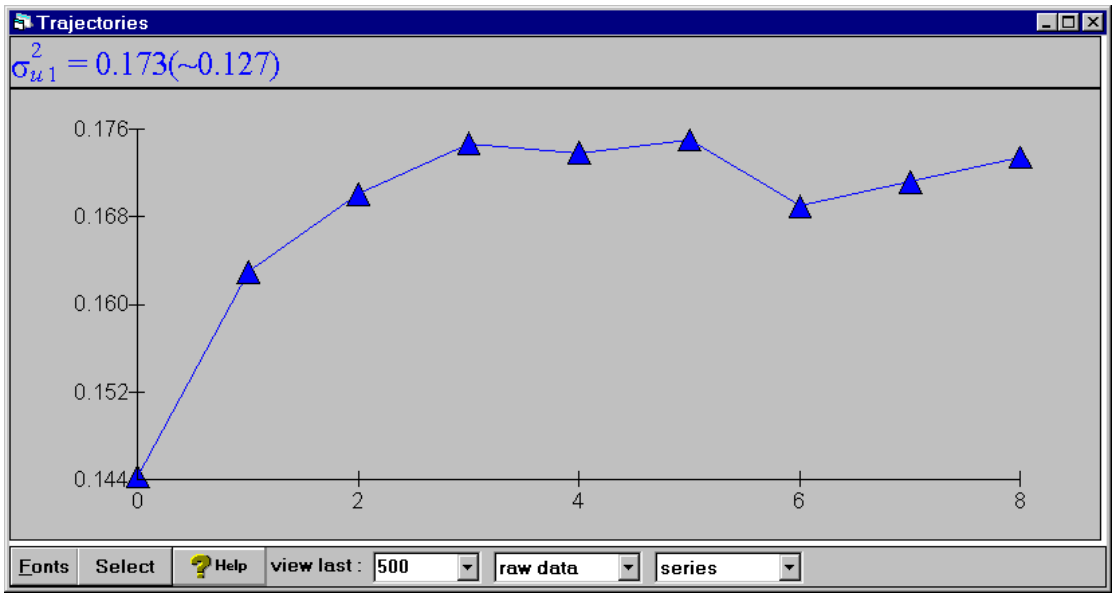


We can again switch between bias corrected and uncorrected estimates by opening the **Estimation control** window and checking the **bias corrected** box. Selecting bias corrected estimates and scaled SE's from the **Estimation control** window will transform the **Trajectories** window as follows:

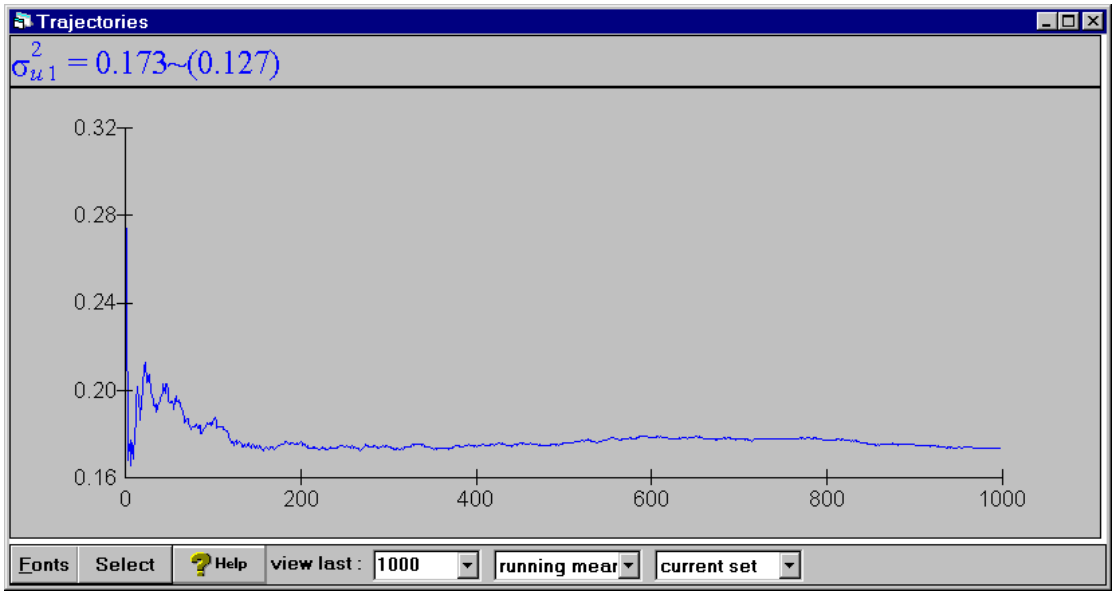


In section 5.3 we saw that running for 5 sets of 100 iterations was not long enough for the parametric bootstrap. A similar result will be seen for the non-parametric. We will skip this here and instead immediately increase the replicate set size to 1000 and the number of replicate sets to 8. Note again that to run the bootstrap for this length of time takes several hours with this model and dataset.

The graph below shows a plot of the (bias corrected) series means for the level 2 variance parameter using non-parametric bootstrapping. This graph compares favourably with the graph for the parametric bootstrap which had final estimate 0.171(~0.124).

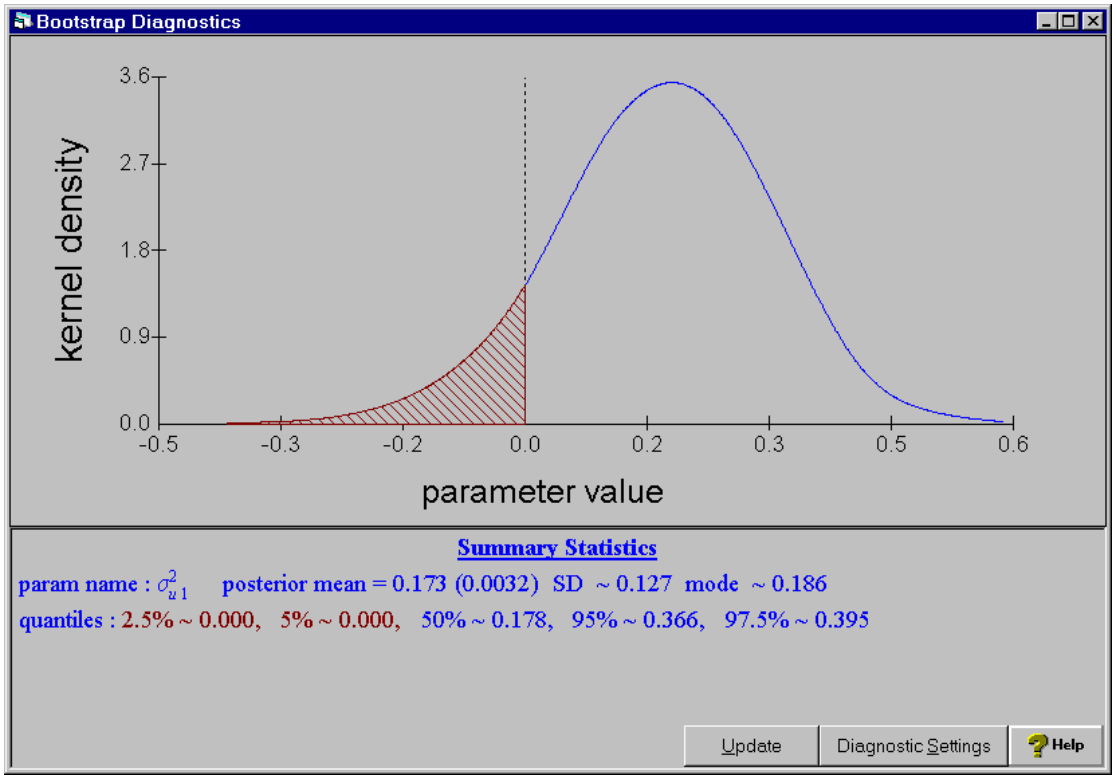


Here we see that the graph looks fairly stable after step 3, and from an original estimate of 0.144 the bootstrap eventually produces a bias corrected estimate of 0.173. The running mean for the last replicate set on this bootstrap is:



We see that this replicate set has stabilised by the time 200 replicates have been reached. This means that for the non-parametric bootstrap on this example we could probably have used a replicate set size of 250 and a series of 5 sets. It is generally sensible, however to select replicate set size and series lengths that are conservative.

To use the last chain to get interval estimates for this parameter we firstly need to **select raw data** instead of **running mean** from the middle pull-down option box on the **Trajectories** window. This will bring up the actual chain of 1,000 bootstrap replicates in the final replicate set. Now clicking on the **Trajectories** window will bring up the **Bootstrap Diagnostics** window as shown below.



Here we see that running for 1,000 replicates produces a smoother curve than seen in the earlier plot for the parametric bootstrap based on 100 replicates. We again see here that both the 2.5% and 5.0% quantiles are estimated as zero as they correspond to negative values in the kernel density plot. This means that for this model the level 2 variance is not significantly different from zero.

What you should have learnt from this chapter

How to use the bootstrap options in *MLwiN*

How the iterative bootstrap gives unbiased estimates

The difference between parametric and non-parametric bootstrapping.

Chapter 18: Modelling cross-classified data units using the Command Interface

The motivation for multilevel modelling is that most social processes we wish to model take place in the context of a hierarchical social structure. The assumption that social structures are purely hierarchical, however, is often an over-simplification. People may belong to more than one grouping at a given level of a hierarchy and each grouping can be a source of random variation. For example, in an educational context both the neighbourhood a child comes from and the school a child goes to may have important effects. A single school may contain children from many neighbourhoods and different children from any one neighbourhood may attend several different schools. Therefore school is not nested within neighbourhood and neighbourhood is not nested within school: we have a cross-classified structure. The consequences of ignoring an important cross-classification are similar to those of ignoring an important hierarchical classification.

A simple model in this context can be written:

$$y_{i(jk)} = \alpha + u_j + u_k + e_{i(jk)} \qquad (1)$$

where the achievement score $y_{i(jk)}$ of the *i*th child from the (*jk*)th school/neighbourhood combination is modelled by the overall mean α, together with a random departure u_j due to school *j*, a random departure u_k due to neighbourhood *k*, and an individual-level random departure $e_{i(jk)}$.

The model can be elaborated by adding individual-level explanatory variables, whose coefficients may also be allowed to vary across schools or neighbourhoods. Also, school or neighbourhood level variables can be added to explain variation across schools or neighbourhoods.

Another example occurs when each pupil takes a single exam paper which is assessed by a set of raters. If a different set of raters operates in each school we have a pupil/rater cross-classification at level 1 nested within schools at level 2. A simple model for this situation can be written:

$$y_{(ij)k} = \alpha + u_k + e_{ik} + e_{jk}$$

where the rater and pupil effects are modelled by the level 1 random variables e_{ik} and e_{jk}. The cross-classification need not be balanced, and some pupils' papers may not be assessed by all the raters. Another example is where a sample of different measurers measured the weights of a sample of animals, each animal being measured

once. If independent repeat measurements were made by the raters on each animal this would become a level 2 cross classification with replications within cells. In fact we could view the first case as a level 2 classification where there just happened to be only one observation per cell. Many cross classifications will allow such alternative design interpretations.

Returning to the raters, if the same set is used in different schools then raters are cross-classified with schools. An equation such as (1) can be used to model this situation, where now k refers to raters rather than neighbourhoods. If in addition schools are crossed by neighbourhoods, then pupils are nested within a three-way rater/school/neighbourhood classification. For this case we may extend equation (1) by adding a term u_l for the rater classification:

$$y_{i(jkl)} = \alpha + u_j + u_k + u_l + e_{i(jkl)} \qquad (3)$$

If raters are not crossed with schools, but schools are crossed with neighbourhoods, a simple formulation might be:

$$y_{(ij)(kl)} = \alpha + u_k + u_l + e_{i(kl)} + e_{j(kl)} \qquad (4)$$

where now i refers to pupils, j to raters, k to schools, and l to neighbourhoods.

Other applications are found, for example, in survey analysis where interviewers are crossed with areas.

How cross-classified models are implemented in *MLwiN*

Suppose we have a level 2 cross-classification with 100 schools drawing pupils from 30 neighbourhoods. If we sort the data into school order and ignore the cross-classification with neighbourhoods, the schools impose the usual block-diagonal structure on the N by N covariance matrix of responses, where N is the number of students in the data set. To incorporate a random neighbourhood effect we must estimate a non-block-diagonal covariance structure.

We can do this by declaring a third level in our model with one unit which spans the entire data set. We then create 30 dummy variables one for each neighbourhood and allow the coefficients of these to vary randomly at level 3 with a separate variance for each of our 30 neighbourhoods. We constrain all 30 variances to be equal.

We can allow other coefficients to vary randomly across schools by putting them in the model as level 2 random parameters in the usual way. If we wish the coefficient of a covariate - a slope - to vary randomly across neighbourhoods the procedure is more

complicated. We must create 30 new variables which are the product of the neighbourhood dummies and the covariate. These new variables are set to vary randomly at level 3. If we wish to allow intercept and slope to covary across neighbourhoods, we require 90 random parameters at level 3: an intercept variance, a slope variance and an intercept/slope covariance for each of the 30 neighbourhoods. As before we constrain the intercept variances, the covariances and the slope variances to produce 3 common estimates by the use of appropriate constraints. The SETX command is provided to automate this procedure.

It is important to realise that although in this example we have set up a 3-level *MLwiN* structure, conceptually we have only a 2-level model, but with neighbourhood and school crossed at level 2. The third level is declared as a device to allow us to estimate the cross-classified structure. The details of this method are given in Rasbash & Goldstein (1994).

Some computational considerations

Cross-classified models can demand large amounts of storage for their estimation and can be computationally intensive. The storage required to run a model, in worksheet cells, is given by:

$$3n_e b + n_f b + \sum_{\ell=1}^{\ell=L} 4r_\ell b + 2br_{\max} \qquad (5)$$

where

b is the number of level 1 units in the largest highest-level unit

n_e is the number of explanatory variables

n_f is the number of fixed parameters

L is the number of levels in the model

r_ℓ is the number of variances being estimated at level ℓ (covariances add no further space requirements)

$r_{\max}$ is the maximum number of variances at a single level

In cross-classified models n_e will be large, typically the size of the smaller classification, and b will be equal to the size of the entire data set. These facts can lead

to some quite devastating storage requirements for cross-classified models. In the example of 100 schools crossed with 30 neighbourhoods, suppose we have data on 3000 pupils. The storage required to handle the computation for a cross-classified variance components model is:

$$3n_e b + n_f b + \sum_{l=1}^{l=L} 4r_l b + 2br_{\max}$$

$$3*30*3000+3000 + 4(3000+3000+30*3000) + 2*3000*30$$

that is, some 840,000 free worksheet cells. If we wish to model a slope varying across neighbourhoods then the storage requirement doubles.

These storage requirements can be reduced if we can split the data into separate groups of schools and neighbourhoods. For example, if 40 of the 100 schools take children from only 12 of the 30 neighbourhoods, and no child from those 12 neighbourhoods goes to a different school, then we can treat those 40 schools by 12 neighbourhoods as a separate group. Suppose that in this way we can split our data set of 100 schools and 30 neighbourhoods into 3 separate groups, where the first group contains 40 schools and 12 neighbourhoods, the second contains 35 schools and 11 neighbourhoods and the third contains 25 schools and 7 neighbourhoods. We can then sort the data on school within group and make group the third level. We can do this because there is no link between the groups and all the covariances we wish to model are contained within the block diagonal matrix defined by the group blocks.

For a cross-classified variance components model n_e is now the maximum number of neighbourhoods in a group, that is 12, and *b* is the size of the largest group, say 1800. This leads to storage requirements of:

*3*12*1800+1800+4(1800+1800+12*1800)+2*1800*12*

that is, about 210,000 free worksheet cells.

Finding such groupings not only decreases storage requirements but also significantly improves the speed of estimation. The command XSEArch is designed to find such groups in the data.

Modelling a 2-way classification - an example

In this example we analyse children's overall exam attainment at age sixteen. The children are cross-classified by the secondary school and the primary school that they attended. The model is of the form given in equation (16.1) where u_j is a random departure due to secondary school and u_k is a random departure due to primary school.

The data are on 3,435 children who attended 148 primary schools and 19 secondary schools in Fife, Scotland.

The initial analysis requires a worksheet size of just under 1000k cells and this is the default size for *MLwiN*. Note that the default size can be changed from the **Options** menu.

Retrieve the worksheet `XC.WS`. Opening the **Names** window shows that the worksheet contains 11 variables:

`VRQ`	A verbal reasoning score resulting from tests pupils took when they entered secondary school.
`ATTAIN`	Attainment score of pupils at age sixteen.
`PID`	Primary school identifying code
`SEX`	Pupil's gender
`SC`	Pupil's social class
`SID`	Secondary school identifying code
`FED`	Father’s education
`CHOICE`	Choice number of secondary school attended
`MED`	Mother’s education
`CONS`	Constant vector
`PUPIL`	Pupil identifying code

In the following description we shall be using the **Command interface** to set up and fit the models. In later versions of *MLwiN* a graphical user interface will be available.

A 2-level variance components model with primary school at level 2 is already set up. To add the secondary school cross-classification we need to create a level-3 unit spanning the entire data set, create the secondary school dummies, enter them as random parameters at level 3 and create a constraint matrix to pool the 20 separate estimates of the secondary school variances into one common estimate. Apart from declaring the third level which is achieved by typing

```
IDEN 3 'CONS'
```

the remaining operations are all performed by the SETX command which is given at the end of this chapter.

The first component of this command is the set of columns with random coefficient at the pseudo-level introduced to accommodate the cross-classification, and in our case is `CONS`. The psuedo-level (3) comes next. Then comes the column containing the identifying codes for the non-hierarchical classification, which in our case is `SID`.

The SETX command requires the identifying codes for the cross-classified categories to be consecutive and numbered from 1 upwards. If this is not the case identifying codes can be put into this format using the MLREcode command, for example:

```
NAME C12 'NEWSID'

MLREcode 'CONS' 'SID' 'NEWSID'
```

Our secondary and primary identifying codes are already in the correct format so we do not need to use the **MLRE**code command.

The dummies for the non-hierarchical classification are written to the next set of columns. The dummies output are equal in number to *k***r* where *k* is the number of variables in <*explanatory variable group*> and *r* is the number of identifying codes in <*ID column*>. They are ordered by identifying code within explanatory variable. The constraint matrix is written to the last column. In addition the command sets up the appropriate random parameter matrix at level 3.

To set up our model the command is

```
SETX 'CONS' 3 'SID' C101-C119 C20
```

If you examine the setting screen by typing the command **SETT** you will see in the **output** window that the appropriate structures have been created. Before running the model we must activate the constraints by typing

```
RCON C20
```

The model will take some time to run. Four iterations are required to reach convergence and on a 90-MHz Pentium each iteration takes 10 seconds. The results are:

Parameter	Description	Estimate(se)
σ^2_{uj}	between primary school variance	1.12(0.20)
σ^2_{uk}	between secondary school variance	0.35(0.16)
σ^2_e	between individual variance	8.1(0.2)
α	mean achievement	5.50(0.17)

This analysis shows that the variation in achievement at age sixteen attributable to primary school is three times greater than the variation attributable to secondary school. This type of finding is an intriguing one for educational researchers and raises many further issues for study.

Explanatory variables can be added to the model in the usual way to attempt to explain the variation. We must be careful if we wish to create contextual secondary school variables using the ML** family of commands or the equivalent in the **Multilevel data manipulation** window. The data currently are sorted by primary school not secondary school as these manipulations require. Therefore the data must be sorted by secondary school, the contextual variable created, and the data re-sorted by primary school.

Other aspects of the SETX command

When more than one coefficient is to be allowed to vary across a non-hierarchical classification in some circumstances you may not wish the covariance between the coefficients to be estimated. This can be achieved most easily by using two successive SETX commands. The following three examples illustrate.

Example 1

```
SETX 'CONS' 3 'SID' C101-C119 C20
```

This sets up the data for the cross-classified variance components model we have just run.

Example 2

Assuming the model is set up as in example 1 and the constraint matrix is activated, if we now type

```
SETX 'VRQ' 3 'SID' C121-C139 C20
```

we shall have the structure for estimating the variance of the coefficients of VRQ and CONS across secondary schools. The intercept/slope covariance will not be estimated. If you want to run this model you will be told that you need to increase the worksheet size.

Example 3

If no cross-classified structure has yet been specified and we type

```
SETX 'CONS' 'VRQ' 3 'SID' C101-C119 C121-C139 C20
```

we shall have the structure for estimating the variance *and covariance* of the coefficients of VRQ and CONS across secondary schools. If you wish to issue this SETX command following the previous analysis you must first remove all the explanatory dummy variables (see below).

The SETX command adds the constraints it generates to any existing constraints specified with the RCON command. Any additional random-parameter constraints which the user wants must be activated by RCON before issuing any SETX command. In particular, when elaborating cross-classified models with more than one SETX command, you must be sure to activate the constraint column generated by the first SETX before issuing second and subsequent SETX commands. Failure to do so will cause the first set of constraints not to be included in the constraints output by the second SETX command.

One limitation of the SETX command is that it will fail if any of the dummy variables it generates are already in the explanatory variable list. One situation where this may occur is when we have just estimated a cross-classified variance components model and we wish to expand the model to a cross-classified random coefficient regression with slope/intercept covariances estimated. In this case typing

```
SETX 'CONS' 'VRQ' 3 'SID' C101-C119 C121-C139 C20
```

will produce an error message since C101-C119 will already be in the explanatory variable list. The problem can be avoided by removing C101-C119 and then giving the SETX command :

```
EXPL 0 C101-C119
```

```
SETX 'CONS' 'VRQ' 3 'SID' C101-C119 C121-C139 C20
```

Note that if the random constraint matrix is left active, the above EXPL 0 command will remove from the matrix the constraints associated with `C101-C119`, leaving only those which the user had previously specified.

After a SETX command, estimation must be restarted using the STARt command or button.

Reducing storage overhead by grouping

We can increase speed and reduce storage requirements by finding separate groups of secondary/primary schools as described above. The XSEArch command will do this. It has the following format:

Retrieve the original worksheet in XC.WS. We search the data for separated groups by typing

```
XSEArch 'PID' 'SID' C13 C14
```

Looking at C13, the column of separated groups produced, in the **Names** window, we see that it is a constant vector. That is, no separation can be made and all primary and secondary schools belong to one group. The new category codes in C14 therefore span the entire range (1 to 19) of categories in the original non-hierarchical classification. This is not surprising since many of the cells in the 143 by 19 table contain very few individuals. It is this large number of almost empty cells that makes separation impossible. In many circumstances we may be prepared to sacrifice some information by omitting cells with very few students. We can omit data for cells with less than a given number of individuals using the XOMIt command.

The XOMIt command has the format:

In our case we can omit cells containing 2 or fewer members by typing

```
XOMIt 2 C3 C6 C1-C2 C4-C5 C7-C11 C3 C6 C1-C2 C4-C5 C7-C11
```

If we now repeat the XSEArch command exactly as before, we find that `C13`, the group code column, has a range from 1 to 6 indicating that 6 groups have been found. The new category codes have a range from 1 to 8 indicating that the maximum number of secondary schools in any group is 8. Use the **Tabu**late window to produce tables of secondary school and primary school by group. This confirms that with our reduced data set no primary school or secondary school crosses a group boundary.

We now sort the data by primary school within separated group. The group codes are now used to define a block diagonal structure for the variance-covariance matrix at level 3 which reduces the storage overhead and speeds up estimation. The following commands set up the model (or the **Sort** window can be used).

```
SORT 2 c13 c3 C1 C2 C4-C11 C14 C13 C3 C1 C2 C4-C11 C14

IDEN 3 C13

SETX 'CONS' 3 C14 C101-C108 C20

RCON C20
```

Notice that the new category codes in `C14` running from 1 to 8 (the maximum number of secondary schools in a separated group) are now used as the category codes for the non-hierarchical classification. This means we now need only 8 as opposed to 19 dummies to model this classification.

Estimation proceeds more than four times faster than in the full model, with very similar results.

Parameter	Description	Estimate(se)
σ^2_{uj}	between primary school variance	1.10(0.20)
σ^2_{uk}	between secondary school variance	0.38(0.19)
σ^2_e	between individual variance	8.1(0.2)
α	mean achievement	5.58(0.18)

Modelling a multi-way classification

The commands described above can also be used to model multi-way classifications. For example, our secondary school by primary school cross-classification could be further crossed, say by neighbourhoods, if neighbourhood identification was available.

In general we can model an *n*-way classification by repeated use of the XSEArch command to establish a separated group structure and then repeated use of the SETX command to specify each classification.

MLwiN commands for cross classifications

The commands used are as follows and are also described in the MLwiN help system:

XOMIt

XOMIt cells with not more than <*value*> members from the cross-classification defined by <*input column-1*> and <*input column-2*> {carrying data in <*input data group*>} results to <*output column-1*> <*output column-2*> {and carried data to <*output data group*>}

XSEArch

XSEArch for separable groups in the cross-classification defined by <*column-1*> and <*column-2*> putting separated group codes in <*group ID column*> and new categories in <*new ID column*>

The first two columns describe the cross-classification to be searched. The non-hierarchical classification is specified by <*column-2*>. If separable groups can be found they are assigned group codes 1, 2, etc. and these are placed in <*group ID column*>. The category codes of <*column-2*> are then recoded in <*new ID column*> to run from 1 within each group. An example will make this clear.

SETX

SETX set a random cross-classification, with coefficients of <*explanatory variable group*> random at level <*value*> across categories in <*ID column*>, storing dummies in <*output group*> and constraints in <*constraints column*>

<*explanatory variable group*> specifies the variables whose coefficients we wish to vary randomly across the non-hierarchical classification, in our case, the secondary schools. Here we wish to run a variance components model so only `CONS` varies across secondary schools.

What you will have learnt from this chapter

- What a cross classification is
- How to set up and fit a cross classified model

Chapter 19: Multiple membership models

Multiple membership models describe the situation where, say in a 2 level model, level 1 units belong to two or more higher level units. Thus, for example, in a longitudinal study of school students, many will change their schools and thus 'belong' to more than one school during the study. When modelling such data the level 2 effect will be shared between all the units to which a student belongs. We therefore need to allocate a set of weights for each student to attach to these units.

A simple multiple membership model

A simple variance components model of this kind a model can be written as follows:

Suppose that we know, for each individual, the weight π_{ij_2}, associated with the j_2-th secondary school for student i (for example the proportion of time spent in that school) with $\sum_{j_2=1}^{J_2} \pi_{ij_2} = 1$.

$$y_{i(j_2)} = (X\beta)_{i(j_2)} + \sum_{j_2} u_{j_2}^{(2)} \pi_{ij_2} + e_{i(j_2)}$$

$$\sum_{j_2} u_{j_2}^{(2)} \pi_{ij_2} = \pi_i u^{(2)}$$

$$u^{(2)^T} = \{u_1^{(2)}, ...u_{J_2}^{(2)}\}$$

$$\pi = \{\pi_1, ...\pi_{J_2}\}$$

$$\pi_{j_2}^T = \{\pi_{1j_2}, ...\pi_{Nj_2}\}$$

where N is the total number of students and $\mathrm{u}^{(2)}$ is the $J_2 \times 1$ vector of secondary school effects. This is therefore a 2-level model where the level 2 variation among secondary schools is modelled using the J_2 sets of weights for student i ($\pi_1, ...\pi_{J_2}$) as explanatory variables, with π_{j_2} the N x 1 vector of student weights for the j_2 th secondary school. We have

$$\mathrm{var}(\mathrm{u}_{\mathrm{j}_2}^{(2)}) = \sigma_{\mathrm{u}2}^2, \quad \mathrm{cov}(\mathrm{u}_{\mathrm{j}_1}^{(1)} \mathrm{u}_{\mathrm{j}_2}^{(2)}) = 0$$

$$\mathrm{var}(\sum_{\mathrm{j}_2} \mathrm{u}_{\mathrm{j}_2}^{(2)} \pi_{\mathrm{ij}_2}) = \sigma_{\mathrm{u}2}^2 \sum_{\mathrm{j}_2} \pi_{\mathrm{ij}_2}^2$$

These models can also be extended to deal with cases where higher level unit identifications are missing. For details of these models with an example see Hill and Goldstein (1997).

There are two new commands which together can be used for specifying such multiple membership models. These are WTCOl and ADDM.

To show the use of the WTCOl command, suppose we have a model with pupils nested within schools and we have only one response per pupil. However, some pupils attend more than one school during the study and we know the identities of the schools they attended and have information on how much time they spent in each school.

Similarly to a cross classified model, we create a set of indicator variables one for each school. Where a pupil attends more than one school they require the indicator variable for each school they attended to be multiplied by a weight, which for example could be based upon the proportion of time the pupil spent at that school. The indicator variables for all the schools the pupil did not attend are set to zero. It is this set of weighted indicator variables that is made to have random coefficients at level 2. As with cross classified models, level 2 is set to be a unit spanning the entire data set and the variances of all the indicator variable coefficients are constrained to be equal.

The WTCOl command can be used to create the weighted indicator variables. If we have 100 schools and the maximum number of schools attend by a pupil is 4 then the WTCOl command would be

WTCOl 4, id columns *C1-C4*, weights in *C5-C8*, weighted indicator columns output to *C101-C200*

Suppose pupil 1 attends only school 5, pupil 2 attends schools 8 and 9 with proportions 0.4 and 0.6, pupil 3 attends schools 4,5,8,6 with proportions 0.2,0.4,0.3,0.1; then the id and weight columns for these 3 pupils would contain the data

c1	c2	c3	c4	c5	c6	c7	c8
5	0	0	0	1	0	0	0
8	9	0	0	0.4	0.6	0	0
4	5	8	6	0.2	0.4	0.3	0.1

The first 9 columns of the output for these three children would be

c101	c102	c103	c104	c105	c106	c107	c108	c109
0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0.4	0.6
0	0	0	0.2	0.4	0.1	0	0.3	0

The second command is the ADDM command.

This command adds sets of weighted indicator variables created by the WTCOL command to the random part of the model at level M and generates a constraint matrix which defines the variances estimated for each set of indicators to be equal. It is possible to have more than one set of indicators if you wish to allow several random coefficients to vary across the multiple membership classification.

First consider, a variance components multiple membership model, continuing from the example outlined in the description of the WTCOl command; in this case we need

`ADDM 1` set of indicators at level `2`, in `C101-C200`, constraints to `C20`

If we wished to allow the slope of an X variable, say "PRETEST", in addition to the intercept, to vary randomly across the multiple membership classification, we must then form another set of indicators which are the product of the original indicators and "PRETEST".

To do this :

```
LINK C101-C200 G1          [put original indicators in group 1]
LINK C201-C300 G2          [set up group for interactions]
CALC G2=G1*'PRETEST'   [create interactions]
```

`ADDM 2` sets of indicators at level `2`, first set in `C101-C200`, second set in `C201-C300,` constraints to `C20`

This last command will place the two sets of indicators in the random part of the model as well as associated covariance terms between the two sets and establish the appropriate constraints in C20. Note that the ADDM command will not add its constraints to any existing constraints in the model.

MLwiN commands for multiple membership models

The commands are as follows and are also described in the *MLwiN* help system:

WTCOl

WTCOl <value> id columns <group 1> weights in columns <group 2> weighted indicator columns to <group 3>

ADDM

ADDM <value> sets of indicators at level <value>, first set in <group>, second set in <group> ….., constraints to <column>

What you will have learnt from this chapter

- What a multiple membership model is
- How to specify a multiple membership model in *MLwiN* .

References

Aitkin, M. and Longford, N. (1986). Statistical modelling in school effectiveness studies (with discussion). *Journal of the Royal Statistical Society*, A 149, 1-43.

Atkinson AC (1986). Masking unmasked. *Biometrika*, 73, 533-541

Barnett V and Lewis T (1994). *Outliers in Statistical Data: 3rd edition*. New York: John Wiley.

Belsey DA, Kuh E and Welsch RE (1980). *Regression Diagnostics*. New York: John Wiley.

Browne, W.J. (1998). Applying MCMC Methods to Multi-level models. PhD. Thesis, University of Bath.

Bryk, A. S. and Raudenbush, S. W. (1992). *Hierarchical Linear Models*. Newbury Park, California, Sage

Diggle, P. and Kenward, M. G. (1994). Informative drop-out in longitudinal data analysis. *J. Royal Statistical Society,* C, 43, 49-93 (with discussion).

Efron, B. and Tibshirani, R. (1993). An Introduction to the Bootstrap. New York: Chapman and Hall.

Efron, B. and Tibshirani, R. J. (1993). *An Introduction to the Bootstrap*. New York, Chapman and Hall.

Gelman, A., Roberts, G.O. and Gilks, W.R. (1995). Efficient Metropolis Jumping Rules. In J.M. Bernardo, J.O. Berger, A.P. Dawid, and A.F.M. Smith (Eds.), Bayesian Statistics 5, pp. 599-607. Oxford: Oxford University Press.

Gilks, W. R., Richardson, S. and Spiegelhalter, D. J. (1996). *Markov chain Monte Carlo in practice*. London, Chapman and Hall:

Goldstein H (1995). *Multilevel Statistical Models: 2nd edition*. London : Edward Arnold.

Goldstein H and Healy MRJ (1995). The graphical representation of a collection of means. *J. Royal Statistical Society, A*, **158**: 175-7.

Goldstein, H. (1979). *The design and analysis of longitudinal studies.* London, Academic Press.

Goldstein, H. (1995a). *Multilevel Statistical Models*. London, Edward Arnold: New York, Wiley.

Goldstein, H. (1995b). The multilevel analysis of growth data. In; *Essays on Auxology*. R. Hauspie, G. Lindgren and F. Falkner. Welwyn Garden City, England, Castlemead.

Goldstein, H. (1996). Consistent estimators for multilevel generalised linear models using an estimated bootstrap. *Multilevel Modelling Newsletter* **8**(1): 3-6.

Goldstein, H. and Healy, M. J. R. (1995). The graphical presentation of a collection of means. *J. Royal Statistical Society, A*. **158**: 175-7.

Goldstein, H. and Rasbash, J. (1996). Improved approximations for multilevel models with binary responses. *Journal of the Royal Statistical Society, A*. **159**: 505-13

Goldstein, H. and Spiegelhalter, D. J. (1996). League tables and their limitations: statistical issues in comparisons of institutional performance. *Journal of the Royal Statistical Society, A*. **159**: 385-443

Goldstein, H., Healy, M. J. R. and Rasbash, J. (1994). Multilevel time series models with applications to repeated measures data. *Statistics in Medicine* **13**: 1643-55.

Goldstein, H., Rasbash, J., Yang, M., Woodhouse, G., et al. (1993). A multilevel analysis of school examination results. *Oxford Review of Education* **19**: 425-433

Heath, A., Yang, M. and Goldstein, H. (1996). Multilevel analysis of the changing relationship between class and party in Britain 1964-1992. *Quality and Quantity* **30**: 389-404

Hill, P. W. and Goldstein, H. (1998). Multilevel modelling of educational data with cross classification and missing identification of units. *Journal of Educational and Behavioural statistics* **23**: 117-128.

Langford, I. H. and Lewis, T. (1998). Outliers in multilevel models (with Discussion). *J. Royal Statistical Society, A*, **161**: 121-160

Langford, I. H., Bentham, G. and McDonald, A. (1998). Multilevel modelling of geographically aggregated health data: a case study on malignant melanoma mortality and UV exposure in the European community. *Statistics in Medicine* **17**: 41-58.

Lawrance A. J. (1995). Deletion influence and masking in regression. *J. Royal Statistical Society, B*, **57:** 181-189.

Longford, N. T. (1993). *Random Coefficient Models*. Oxford, Clarendon Press:

McCullagh, P. and Nelder, J. (1989). *Generalised linear models*. London, Chapman and Hall:

Nuttall, D. L., Goldstein, H., Prosser, R. and Rasbash, J. (1989). Differential school effectiveness. *International Journal of Educational Research* **13**: 769-776.

Plewis, I. (1997) *Statistics in Education*. London: Edward Arnold

Raftery, A. E. and Lewis, S. M. (1992). How many iterations in the Gibbs sampler? In *Bayesian Statistics 4* (eds. J. M. Bernardo, J. O. Berger, A. P. Dawid and A. F. M. Smith), pp 765-76. Oxford: Oxford University Press.

Rasbash, J. and Goldstein, H. (1994). Efficient analysis of mixed hierarchical and cross classified random structures using a multilevel model. *Journal of Educational and Behavioural statistics* **19**: 337-50.

Rowe, K. J. and Hill, P. W. (1997). *Simultaneous estimation of multilevel structural equations to model students' educational progress.* Tenth International Congress for School effectiveness and Improvement, Memphis, Tennessee.

Silverman, B.W. (1986). Density Estimation for Statistics and Data Analysis. London: Chapman and Hall

Tizard, B., Blatchford, P., Burke, J., Farquhar, C. and Plewis, I. (1988). *Young Children at school in the Inner City.* Hove, Sussex; Lawrence Erlbaum.

Velleman, P. F. and Welsch, R. E. (1981). Efficient computing of regression diagnostics. *American Statistician*, **35**: 234-242.

Venables, W. N. and Ripley, B. D. (1994). *Modern Applied Statistics with S-Plus*. New York: Springer-Verlag.

Woodhouse, G. and Goldstein, H. (1989). Educational Performance Indicators and LEA league tables. *Oxford Review of Education* **14**: 301-319.

Yang, M., Goldstein, H., Rath, T. and Hill, N. (1999). The use of assessment data for school improvement purposes. *Oxford Review of Education* **25**: 469-483.

Yang, M., Rasbash, J., Goldstein, H. and Barbosa, M. (1999). *MLwiN macros for advanced multilevel modelling;* version 2.0: London, Centre for Multilevel Modelling, Institute of Education.

Index

A

B

C

D

E

F

G

N

O

P

Q

R

S

T

U

V

W

X