MCMC estimation in MLwiN

Version 2.31

by

William J. Browne

Programming by
William J. Browne, Chris Charlton and Jon Rasbash

Updates for later versions by
William J. Browne, Chris Charlton, Mike Kelly and Rebecca Pillinger

Printed 2014

Centre for Multilevel Modelling
University of Bristol

MCMC Estimation in MLwiN version 2.31

ISBN: 978-0-903024-99-0

Printed in the United Kingdom

First printing November 2004

Updated for University of Bristol, October 2005, January 2009, July 2009, August 2011, January 2012, September 2012 and August 2014.

Contents

Acknowledgements

This book would not have been written without the help of many people.

Firstly thanks to Jon Rasbash who has been responsible for the majority of the programming effort in the MLwiN software package over the past 20 years or so, and is also responsible for much of the interface work between my MCMC estimation engine and the rest of the MLwiN package.

Thanks to all my colleagues at the Centre for Multilevel Modelling both now and in the past. In particular thanks to Harvey Goldstein, Jon Rasbash, Fiona Steele, Min Yang and Philippe Mourouga for their comments and advice on the material in the original version of this book.

Thanks to Chris Charlton for his programming effort in the more recent versions of MLwiN. Thanks to Edmond Ng for assistance in updating ealier versions of the book when MLwiN changed and thanks to Michael Kelly and Rebecca Pillinger for LaTeXing and updating the previous version. Thanks to Hilary Browne for her work on the multilevel modelling website that hosts the manuals and software.

The Economic and Social Research Council (ESRC) has provided me personally with funding off and on since 1998 when I started my first post-doctoral position and has provided members of the project team with continuous funding since 1986 and without their support MLwiN and hence this book would not have been possible.

In particular the grant RES-000-23-1190-A entitled “Sample Size, Identifiability and MCMC Efficiency in Complex Random Effect Models” has allowed me to extend the MCMC features in MLwiN and add the final five chapters to this version of the book.

Thanks to my colleagues at Langford and in particular Richard Parker and Sue Hughes for reading through this extended version and pointing out incorrect screen shots and typographic errors. Thanks also to Mousa Golalizadeh for his work on the ESRC grant and to Camille Szmagard for completing my current postdoc team at Langford.

Thanks to David Draper for his support to me as PhD supervisor at the

University of Bath. Thanks for sparking my interest in multilevel modelling and your assistance on the first release of MLwiN.

Thanks to the past attendees of the MLwiN fellows group for their comments and advice. Thanks in no particular order to Michael Healy, Toby Lewis, Alastair Leyland, Alice McLeod, Vanessa Simonite, Andy Jones, Nigel Rice, Ian Plewis, Tony Fielding, Ian Langford, Dougal Hutchison, James Carpenter and Paul Bassett.

Thanks to the WinBUGS project team (of the time of writing the original book) for assistance and advice on the MLwiN to WinBUGS interface and the DIC diagnostic. Thanks to David Spiegelhalter, Nicky Best, Dave Lunn, Andrew Thomas and Clare Marshall.

Finally thanks to Mary, my lovely daughters, Sarah and Helena, my Mum and Dad and my many friends, relatives and colleagues for their love, friendship and support over the years.

To health, happiness and honesty and many more years multilevel modelling!

William Browne, 7th July 2009.

Preface to the 2009, 2011, 2012 and 2014 Editions

I first wrote a book entitled "MCMC estimation in MLwiN" towards the end of my time at the Centre for Multilevel Modelling at the Institute of Education (in 2002). This original work greatly expanded the couple of chapters that appeared in the MLwiN User's Guide and mirrored the material in the User's Guide whilst including additional chapters that contained extensions and features only available via MCMC estimation.

I then spent four and a half years away from the centre whilst working in the mathematics department at the University of Nottingham. For the first few years at Nottingham, aside from minor bug fixing, the MCMC functionality in MLwiN was fairly static. In 2006 I started an ESRC project RES-000-23-1190-A which allowed me to incorporate some additional MCMC functionality into MLwiN. This new functionality does not increase the number of models that can be fitted via MCMC in MLwiN but offers some alternative MCMC methods for existing models.

I needed to document these new features and so rather than creating an additional manual I have added 5 chapters to the end of the existing book which in the interim has been converted to LaTeX by Mike Kelly for which I am very grateful. I also took the opportunity to update the existing chapters a little. The existing chapters were presented in the order written and so I have also taken the opportunity to slightly reorder the material.

The book now essentially consists of 5 parts. Chapters 1-9 cover single level and nested multilevel Normal response models. Chapters 10-13 cover other response types. Chapters 14-17 cover other non-nested structures and measurement errors. Chapters 18-20 cover multivariate response models including multilevel factor analysis models and finally chapters 21-25 cover additional MCMC estimation techniques developed specifically for the latest release of MLwiN.

The book as written can be used with versions of MLwiN from 2.13 onward - earlier versions should work with chapters 1-20 but the new options will not be available. This version also describes the WinBUGS package and the MLwiN to WinBUGS interface in more detail. I used WinBUGS version 1.4.2

when writing this version of the book and so if you use a different version you may encounter different estimates, such is the nature of Monte Carlo estimation and evolving estimation.

Please report any problems you have replicating the analyses in this book and indeed any bugs you find in the MCMC functionality within MLwiN. Happy multilevel modelling!

William J. Browne, 7th July 2009.

This book has been slightly updated for versions of MLwiN from 2.24 onwards. Historically the residuals produced by the IGLS algorithm in MLwiN have been used as starting values when using MCMC. This doesn't really make much sense for models like cross-classified and multiple-membership models where the IGLS estimates are not from the same model. We have therefore made some changes to the way starting values are given to MCMC. As MCMC methods are stochastic the change results in some changes to screen shots in a few chapters. We have also taken this opportunity to correct a few typographical mistakes including a typo in the Metropolis macro in chapter 1 and in the quantiles for the rank2 macro in chapter 4.

William J. Browne, 10th August 2011.

This book has had one further change for version 2.25 onwards with regard residual starting values for models like cross-classified and multiple-membership models. We initially made these all zero but this didn't have the desired effect and so they are now chosen at random from Normal distributions.

William J. Browne, 31st January 2012.

Dedicated to the memory of Jon Rasbash. A great mentor and friend who will be sorely missed.

Chapter 1

Introduction to MCMC Estimation and Bayesian Modelling

In this chapter we will introduce the basic MCMC methods used in MLwiN and then illustrate how the methods work on a simple linear regression model via the MLwiN macro language. Although MCMC methods can be used for both frequentist and Bayesian inference, it is more common and easier to use them for Bayesian modelling and this is what we will do in MLwiN.

1.1 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 previous releases of MLwiN, MCMC estimation has been restricted to a subset of the potential models that can be fitted in MLwiN. This release of MLwiN allows the fitting of many more models using MCMC, including many models that can only be fitted using MCMC but there are still some models where only the maximum likelihood methods can be used and the software will warn you when this is the case.

We will start this chapter with some of the background and theory behind MCMC methods and Bayesian statistics before going on to consider developing the steps of the algorithms to fit a linear regression model. This we will do using the MLwiN macro language. We will be using the same examination dataset that is used in the User's Guide to MLwiN (Rasbash et al., 2008) and in the next chapter we demonstrate how simple linear regression models may be fitted to these data using the MCMC options in MLwiN.

Users of earlier MLwiN releases will find that the MCMC options and screen layouts have been modified slightly and may find this manual 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.

1.2 MCMC methods and Bayesian modelling

We will be using MCMC methods in a Bayesian framework. Bayesian statistics is a huge subject that we cannot hope to cover in the few lines here. Historically Bayesian statistics has been quite theoretical, as until about twenty years or so ago it had not been possible to solve practical problems through the Bayesian approach due to the intractability of the integrations involved. The increase in computer storage and processor speed and the rise to prominence of MCMC methods has however meant that now practical Bayesian statistical problems can be solved.

The Bayesian approach to statistics can be thought of as a sequential learning approach. Let us assume we have a problem we wish to solve, or a question we wish to answer: then before collecting any data we have some (prior) beliefs/ideas about the problem. We then collect some data with the aim of solving our problem. In the frequentist approach we would then take these data and with a suitable distributional assumption (likelihood) we could make population-based inferences from the sample data. In the Bayesian approach we wish to combine our prior beliefs/ideas with the data collected to produce new posterior beliefs/ideas about the problem. Often we will have no prior knowledge about the problem and so our posterior beliefs/ideas will combine this lack of knowledge with the data and will tend to give similar answers to the frequentist approach. The Bayesian approach is sequential in nature as we can now use our posterior beliefs/ideas as prior knowledge and collect more data. Incorporating this new data will give a new posterior belief.

The above paragraph explains the Bayesian approach in terms of ideas, in reality we must deal with statistical distributions. For our problem, we will have some unknown parameters, θ, and we then condense our prior beliefs into a prior distribution, $\mathrm{p}(\theta)$. Then we collect our data, y, which (with a distributional assumption) will produce a likelihood function, $\mathrm{L}(\mathrm{y}|\theta)$, which is the function that maximum likelihood methods maximize. We then combine these two distributions to produce a posterior distribution for θ, $\mathrm{p}(\theta|\mathrm{y}) \propto \mathrm{p}(\theta)\mathrm{L}(\mathrm{y}|\theta)$. This posterior is the distribution from which inferences about θ are then reached. To find the implicit form of the posterior distribution we would need to calculate the proportionality constant. In all but the simplest problems this involves performing a many dimensional integration, the historical stumbling block of the Bayesian approach. MCMC methods however circumvent this problem as they do not calculate the exact form of the posterior distribution but instead produce simulated draws from it.

Historically, the methods used in MLwiN were IGLS and RIGLS, which are likelihood-based frequentist methods. These methods find maximum likelihood (restricted maximum likelihood) point estimates for the unknown parameters of interest in the model. These methods are based on iterative procedures and the process involves iterating between two deterministic steps until two consecutive estimates for each parameter are sufficiently close together, and hence convergence has been achieved. These methods are designed specifically for hierarchical models although they can be adapted to fit other models. They give point estimates for all parameters, estimates of the parameter standard deviations and large sample hypothesis tests and confidence intervals (see the User's Guide to MLwiN for details).

MCMC methods are more general in that they can be used to fit many more statistical models. They generally consist of several distinct steps making it easy to extend the algorithms to more complex structures. They are simulation-based procedures so that rather than simply producing point estimates the methods are run for many iterations and at each iteration an estimate for each unknown parameter is produced. These estimates will not be independent as, at each iteration, the estimates from the last iteration are used to produce new estimates. The aim of the approach is then to generate a sample of values from the posterior distribution of the unknown parameters. This means the methods are useful for producing accurate interval estimates (Note that bootstrapping methods, which are also available in MLwiN can also be used in a similar way).

Let us consider a simple linear regression model

$$\begin{aligned} y_i &= \beta_0 + \beta_1 x_{1i} + e_i \\ e_i &\sim \mathrm{N}(0, \sigma_e^2) \end{aligned}$$

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, and σ_e^2, with the data. As mentioned above 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 and this is known as the *posterior*.

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 distributions. These summaries are currently univariate. From the random draws of a 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 quantiles of this distribution.

In the rest of this chapter, 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 & Spiegelhalter (1996) gives more in-depth material on these topics than is covered here.

1.3 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 this release the following diffuse prior distributions are used (note these are slightly different from the default priors used in release 1.0 and we have modified the default prior for variance matrices since release 1.1):

- For fixed parameters $p(\beta) \propto 1$. This *improper* uniform 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, $p(\frac{1}{\sigma^2}) \sim \Gamma(\varepsilon, \varepsilon)$, where ε is very small. This (proper) prior is more or less equivalent to a Uniform prior for $\log(\sigma^2)$.
- For variance matrices $p(\Omega^{-1}) \sim \text{Wishart}_p(p, p, \hat{\Omega})$ where p 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 these 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. We will show you in a later chapter how to write a simple macro to carry out a simple simulation in MLwiN. The priors used in release 1.0 and informative priors can also be specified and these will be discussed in later chapters. Note that in this development release the actual priors used are displayed in the Equations window.

1.4 MCMC estimation

The models fitted in MLwiN contain many unknown parameters of interest, and the objective of using MCMC estimation for these models is to generate a sample of points in the space defined by the *joint posterior* of these parameters. In the simple linear regression model defined earlier we have three unknowns, and our aim is to generate samples from the distribution $p(\beta_0, \beta_1, \sigma_e^2|y)$. Generally to calculate the joint posterior distribution directly will involve integrating over many parameters, which in all but the simplest examples proves intractable. Fortunately, however, 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. It can be shown that sampling from these conditional posterior distributions in turn is equivalent to sampling from the *joint posterior distribution.*

1.5 Gibbs sampling

The first MCMC method we will consider is *Gibbs Sampling.* Gibbs sampling works by simulating a new value for each parameter (or block of parameters) in turn from its conditional distribution assuming that the current values for the other parameters are the true values. For example, consider again the linear regression model.

We have here three unknown variables β_0, β_1 and σ_e^2 and we will here consider updating each parameter in turn. Note that there is lots of research in MCMC methodology involved in finding different blocking strategies to produce less dependent samples for our unknown parameters (Chib & Carlin, 1999; Rue, 2001; Sargent et al., 2000) and we will discuss some such methods in later chapters.

Ideally if we could sample all the parameters together in one block we would have independent sampling. Sampling parameters individually (often called single site updating) as we will describe here will induce dependence in the chains of parameters produced due to correlations between the parameters. Note that in the dataset we use in the example, because we have centred both the response and predictor variables, there is no correlation between the intercept and slope and so sampling individually still gives independent chains. In MLwiN as illustrated in the next chapter we actually update all the fixed effects in one block, which reduces the correlation.

Note that, given the values of the fixed parameters, the residuals e_i can be calculated by subtraction and so are not included in the algorithms that follow.

First we need to choose starting values for each parameter, $\beta_0(0)$, $\beta_1(0)$ and $\sigma_e^2(0)$, and in MLwiN these are taken from the current values stored 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

1. $p(\beta_0|y, \beta_1(0), \sigma_e^2(0))$ to generate $\beta_0(1)$, and then from
2. $p(\beta_1|y, \beta_0(1), \sigma_e^2(0))$ to generate $\beta_1(1)$, and then from
3. $p(\sigma_e^2|y, \beta_0(1), \beta_1(0)$ to generate $\sigma_e^2(1)$.

Having performed all three 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.

To calculate point and interval estimates from a *Markov chain* we assume that its values are a sample from the posterior distribution for the parameter it represents. We can then construct any summaries for that parameter that we want, for example the sample mean can easily be found from the chain and we can also find quantiles, e.g. the median of the distribution by sorting the data and picking out the required values.

As we have started our chains off at particular starting values it will generally take a while for the chains to settle down (converge) and sample from the actual posterior distribution. The period when the chains are settling down is normally called the burn-in period and these iterations are omitted from the sample from which summaries are constructed. The field of MCMC convergence diagnostics is concerned with calculating when a chain has converged to its equilibrium distribution (here the joint posterior distribution) and there are many diagnostics available (see later chapters). In MLwiN by default we run for a burn-in period of 500 iterations. As we generally start from good starting values (ML estimates) this is a conservative length and we could probably reduce it.

The Gibbs sampling 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. In our example we have three conditional distributions to calculate.

To calculate the form of the conditional distribution for one parameter we write down the equation for the conditional posterior distribution (up to proportionality) and assume that the other parameters are known. The trick is then that standard distributions have particular forms that can be matched to the conditional distribution, for example if x has a Normal(μ, σ^2) distribution then we can write: $p(x) \propto \exp(ax^2 + bx + \mathit{const})$, where $a = -\frac{1}{2\sigma^2}$ and $b = \frac{\mu}{\sigma^2}$, so we are left to match parameters as we will demonstrate in the example that follows.

Similarly if x has a $\Gamma(\alpha, \beta)$ distribution then we can write: $p(x) \propto x^a \exp(bx)$, where $a = \alpha - 1$ and $b = -\beta$.

We will assume here the MLwiN default priors, $p(\beta_0) \propto 1$, $p(\beta_1) \propto 1$, $p(1/\sigma_e^2) \sim \Gamma(\varepsilon, \varepsilon)$, where $\varepsilon = 10^{-3}$. Note that in the algorithm that follows we work with the precision parameter, $1/\sigma_e^2$, rather than the variance, σ_e^2, as it has a distribution that is easier to simulate from. Then our posterior distributions can be calculated as follows

Step 1: β_0

$$\begin{aligned}
p(\beta_0|y, \beta_1, \sigma_e^2) &\propto \prod_i \left(\frac{1}{\sigma_e^2}\right)^{1/2} \exp\left[-\frac{1}{2\sigma_e^2}(y_i - \beta_0 - x_i\beta_1)^2\right] \\
&\propto \exp\left[-\frac{N}{2\sigma_e^2}\beta_0^2 + \frac{1}{\sigma_e^2}\sum_i (y_i - x_i\beta_1)\beta_0 + \mathit{const}\right] \\
&= \exp\left[a\beta_0^2 + b\beta_0 + \mathit{const}\right]
\end{aligned}$$

Matching powers gives:

$$\sigma_{\beta_0}^2 = -\frac{1}{2a} = \frac{\sigma_e^2}{N} \quad \text{and} \quad \mu_{\beta_0} = b\sigma_{\beta_0}^2 = \frac{1}{N}\sum_i (y_i - x_i\beta_0),$$

$$\text{and so} \quad p(\beta_0|y, \beta_1, \sigma_e^2) \sim \mathrm{N}\left(\frac{1}{N}\sum_i (y_i - x_i\beta_1), \frac{\sigma_e^2}{N}\right)$$

Step 2: β_1

$$p(\beta_1|y,\beta_0,\sigma_e^2) \propto \prod_i \left(\frac{1}{\sigma_e^2}\right)^{1/2} \exp\left[-\frac{1}{2\sigma_e^2}(y_i-\beta_0-x_i\beta_1)^2\right]$$
$$\propto \exp\left[-\frac{1}{2\sigma_e^2}\sum_i x_i^2\beta_1^2 + \frac{1}{\sigma_e^2}\sum_i (y_i-\beta_0)x_i\beta_1 + const\right]$$

Matching powers gives:

$$\sigma_{\beta_1}^2 = -\frac{1}{2a} = \frac{\sigma_e^2}{\sum\limits_i x_i^2} \quad \text{and} \quad \mu_{\beta_1} = b\sigma_{\beta_1}^2 = \frac{\sum\limits_i (y_i-\beta_0)x_i}{\sum\limits_i x_i^2},$$
$$\text{and so} \quad p(\beta_1|y,\beta_0,\sigma_e^2) \sim N\left(\frac{\sum\limits_i y_i x_i - \beta_0\sum\limits_i x_i}{\sum\limits_i x_i^2}, \frac{\sigma_e^2}{\sum\limits_i x_i^2}\right)$$

Step 3: $1/\sigma_e^2$

$$p\left(\frac{1}{\sigma_e^2}|y,\beta_0,\beta_1\right) \propto \left(\frac{1}{\sigma_e^2}\right)^{\varepsilon-1} \exp\left[-\frac{\varepsilon}{\sigma_e^2}\right] \prod_i \left(\frac{1}{\sigma_e^2}\right)^{1/2} \exp\left[-\frac{1}{2\sigma_e^2}(y_i-\beta_0-x_i\beta_1)^2\right]$$
$$\propto \left(\frac{1}{\sigma_e^2}\right)^{\frac{N}{2}+\varepsilon-1} \exp\left[-\frac{1}{\sigma_e^2}\left(\varepsilon + \frac{1}{2}\sum_i (y_i-\beta_0-x_i\beta_1)^2\right)\right]$$
$$\text{and so} \quad p\left(\frac{1}{\sigma_e^2}|y,\beta_0,\beta_1\right) \sim \Gamma\left(\varepsilon + \frac{N}{2}, \varepsilon + \frac{1}{2}\sum_i e_i^2\right)$$

So in this example we see that we can perform one iteration of our Gibbs sampling algorithm by taking three random draws, two from Normal distributions and one from a Gamma distribution. It is worth noting that the first two conditional distributions contain summary statistics, such as $\sum\limits_i x_i^2$, which are constant throughout the sampling and used at every iteration. To simplify the code and speed up estimation it is therefore worth storing these summary statistics rather than calculating them at each iteration. Later in this chapter we will give code so that you can try running this model yourself.

1.6 Metropolis Hastings sampling

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 new estimate for the next iteration or rejected and the current value is used as the new estimate for 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.

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 at time step t say, has the property that it is symmetric in $\theta(t-1)$ and $\theta(t)$, that is:

$$p(\theta(t) = a|\theta(t-1) = b) = p(\theta(t) = b|\theta(t-1) = a)$$

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:

1. Draw β_0^* from the proposal distribution $\beta_0(t) \sim \mathrm{N}(\beta_0(t-1), \sigma_p^2)$ where σ_p^2 is the proposal distribution variance.
2. Define $r_t = p(\beta_0^*, \beta_1(t-1), \sigma_e^2(t-1)|y)/p(\beta_0(t-1), \beta_1(t-1), \sigma_e^2(t-1)|y)$ as the posterior ratio and let $a_t = \min(1, r_t)$ be the acceptance probability.
3. Accept the proposal $\beta_0(t) = \beta_0^*$ with probability a_t, otherwise let $\beta_0(t) = \beta_0(t-1)$

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 later chapters.

1.7 Running macros to perform Gibbs sampling and Metropolis Hastings sampling on the simple linear regression model

MLwiN is descended from the DOS based multilevel modelling package MLn which itself was built on the general statistics package Nanostat written by Professor Michael Healy. The legacy of both MLn and Nanostat lives on in MLwiN within its macro language. Most functions that are performed via selections on the menus and windows in MLwiN will have a corresponding command in the macro language. These commands can be input directly into MLwiN via the **Command interface** window available from the **Data Manipulation** menu. The list of commands and their parameters are covered in the Command manual (Rasbash et al., 2000) and in the interactive help available from the **Help** menu.

The user can also create files of commands for example to set up a model or run a simulation as we will talk about in Chapter 8. These files can be created and executed via the macros options available from the **File** menu. Here we will look at a file that will run our linear regression model on the **tutorial** dataset described in the next chapter.

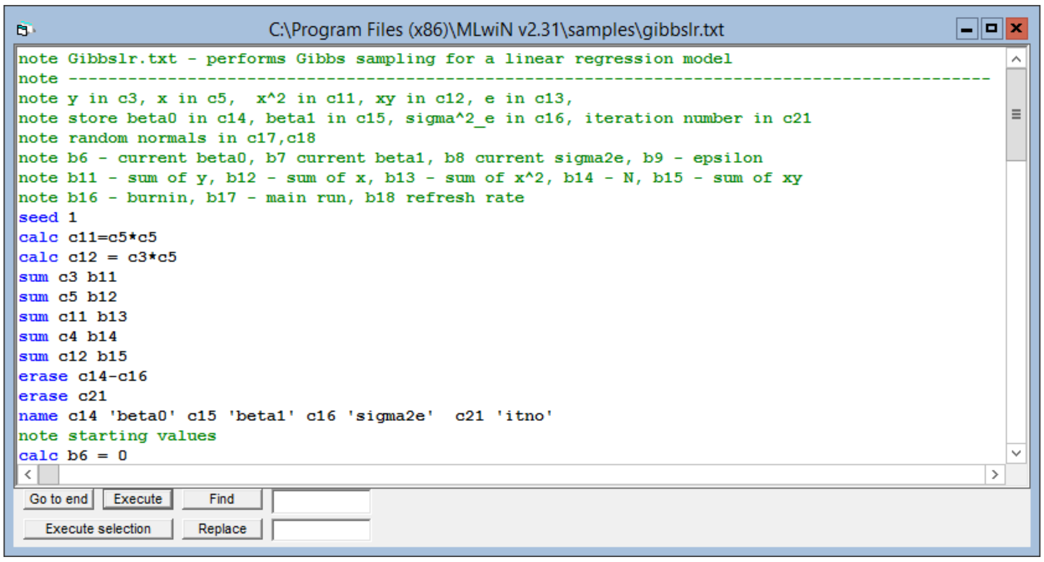
We will firstly have to load up the tutorial dataset:

- Select **Open Sample Worksheet** from the **File** menu.
- Select **tutorial.ws** from the list of possible worksheets.

When the worksheet is loaded its name (plus filepath) will appear at the top of the screen and the **Names** window will appear giving the variable names in the worksheet. We now need to load up the macro file:

- Select **Open Macro** from the **File** menu.
- Select **gibbslr.txt** from the list of possible macros.

When the macro has been loaded a macro window showing the first twenty or so lines of the macro will appear on the screen:



C:\Program Files (x86)\MLwiN v2.31\samples\gibbslr.txt

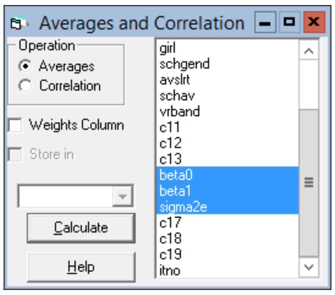
```
note Gibbslr.txt - performs Gibbs sampling for a linear regression model
note --------------------------------------------------------------------------------------------
note y in c3, x in c5,  x^2 in c11, xy in c12, e in c13,
note store beta0 in c14, beta1 in c15, sigma^2_e in c16, iteration number in c21
note random normals in c17,c18
note b6 - current beta0, b7 current beta1, b8 current sigma2e, b9 - epsilon
note b11 - sum of y, b12 - sum of x, b13 - sum of x^2, b14 - N, b15 - sum of xy
note b16 - burnin, b17 - main run, b18 refresh rate
seed 1
calc c11=c5*c5
calc c12 = c3*c5
sum c3 b11
sum c5 b12
sum c11 b13
sum c4 b14
sum c12 b15
erase c14-c16
erase c21
name c14 'beta0' c15 'beta1' c16 'sigma2e'  c21 'itno'
note starting values
calc b6 = 0
```

You will notice that the macro contains a lot of lines in green beginning with the word **note** and this command is special in that it is simply a comment used to explain the macro code and does nothing when executed. The macro sets up starting values and then loops around the 3 steps of the Gibbs sampling algorithm as detailed earlier for the number of stored iterations (b17) plus the length of the burn-in (b16).

To run the macro we simply press the **Execute** button on the macro window. The mouse pointer will turn into an egg timer while the macro runs and then back to a pointer when the macro has finished. The chains of values for the three parameters have been stored in columns c14–c16 and we can look at some summary statistics via the **Averages and Correlations** window

- Select **Averages and Correlations** from the **Basic Statistics** menu

If we now scroll down the list of columns we can select the three output columns that contain the chains, these have been named **beta0**, **beta1** and **sigma2e**. Note to select more than one column in this and any other window press the 'Ctrl' key when you click on the selection with the mouse. When the three are selected the window should look as follows:



Averages and Correlation

Operation: [✓ Averages] Correlation

Weights Column

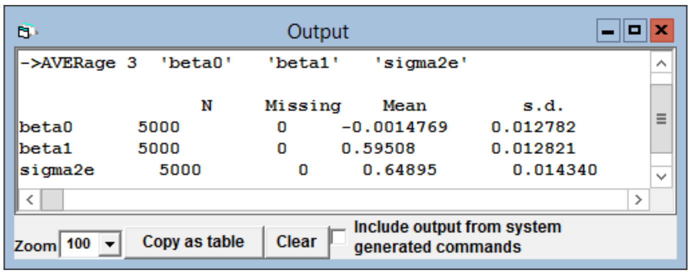
Store in

girl
schgend
avslrt
schav
vrband
c11
c12
c13
[✓ beta0]
[✓ beta1]
[✓ sigma2e]
c17
c18
c19
itno

Now to display the estimates:

- Click the **Calculate** button

and the output window will appear with the following estimates:



```
->AVERage 3  'beta0'    'beta1'    'sigma2e'

                  N      Missing     Mean         s.d.
beta0       5000          0      -0.0014769     0.012782
beta1       5000          0      0.59508        0.012821
sigma2e        5000          0      0.64895        0.014340
```

These estimates are almost identical to those produced by the MLwiN MCMC engine. Any slight differences will be due to the stochastic nature of MCMC algorithms and will reduce as the number of updates is increased.

1.8 Dynamic traces for MCMC

One feature that is offered in MLwiN and some other MCMC based packages such as WinBUGS (Spiegelhalter et al., 2000*a*) is the ability to view estimate traces that update as the estimation proceeds. We can perform a crude version of this with our macro code that we have written to fit this model. If you scan through the code you will notice that we define a box b18 to have value 50 and describe this in the comments as the refresh rate. Near the bottom of the code we have the following switch statement:

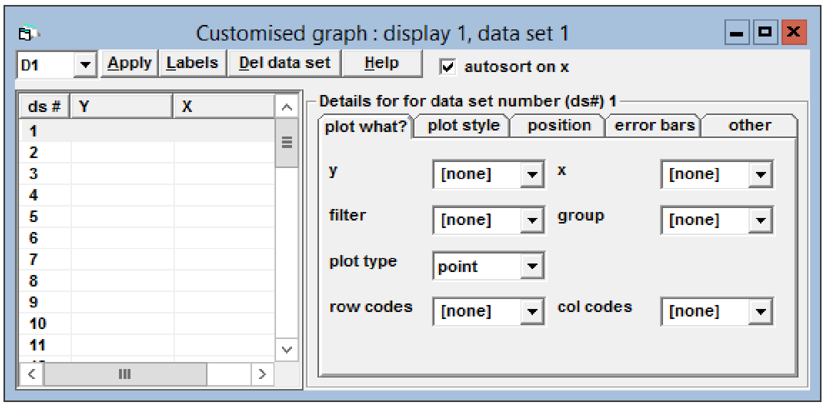
```
calc b60 = b1 mod b18
  switch b60
    case 0:
      pause 1
      leave
    ends
```

The box b1 stores the current iteration and all this switch statement is really saying is if the iteration is a multiple of 50 (b18) perform the **pause 1** command. The **pause 1** command simply releases control of MLwiN from the macro for a split second so that all the windows can be updated. This will be how we set up dynamic traces and we will use this command again in the simulation chapter later.

We now have to set up the graphs for the traces. The Customised graph window is covered in reasonable detail in Chapter 5 of the User's Guide to MLwiN and so we will abbreviate our commands here for brevity. Firstly:

- Select the **Customised Graph(s)** option from the **Graph** menu

This will bring up the blank Customised graph window:



We will now select three graphs (one for each variable).

- Select **beta0** from the **y** list
- Select **itno** from the **x** list
- Select **line** from the **plot type** list

This will set up the first graph (although not show it yet). We now need to add the other two graphs:

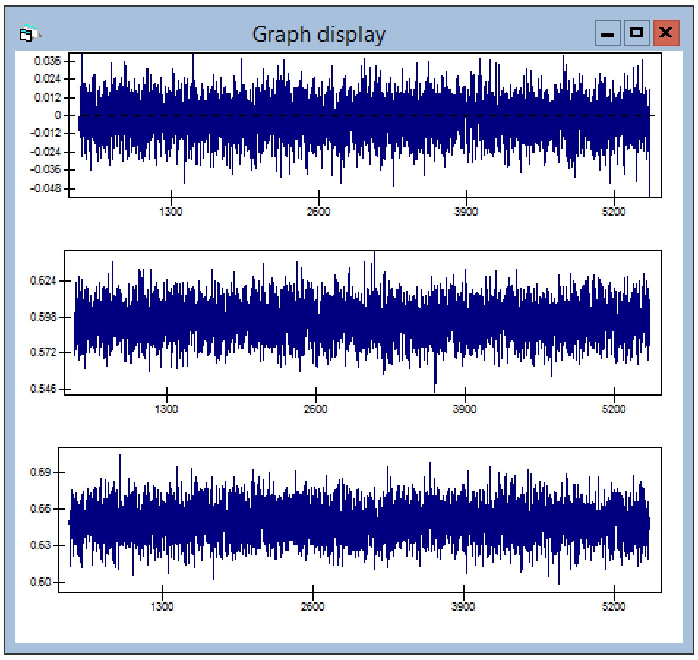
- Select **ds#2** (click in **Y** box next to **2**) on the left of the screen.
- If this is done correctly the settings for all the **plot what?** tabs will reset.
- Select **beta1** from the **y** list.
- Select **itno** from the **x** list.
- Select **line** from the **plot type** list.
- Now select the **position** tab.
- Click in the second box in the first column of the grid.
- If this is done correctly the initial X will vanish and appear in this new position.

Finally for parameter 3:

- Select **ds#3** (click in **Y** box next to **3**) on the left of the screen.
- Select **sigma2e** from the **y** list.

- Select **itno** from the **x** list.
- Select **line** from the **plot type** list.
- Now select the **position** tab.
- Click in the third box in the first column of the grid.
- Click on **Apply** and the 3 graphs will be drawn.

As we have already run the Gibbs sampler we should get three graphs of the 5000 iterations for these runs as follows:



These chains show that the Gibbs sampler is mixing well as the whole of the posterior distribution is being visited in a short period of time. We can tell this by the fact that there are no white large white patches on the traces. Convergence and mixing of Markov chains will be discussed in later chapters.

If we wish to now have dynamic traces instead we can simply restart the macro by pressing the **Execute** button on the macro window. Note that as the iterations increase estimation will now slow down as the graphs redraw all points every refresh! Note also that after the chains finish you will get the same estimates as you had for the first run. This is because the macro has a **Seed** command at the top. This command sets the MLwiN random number seed used and although the MCMC estimation is stochastic, given the same parameter starting values and random numbers it is obviously deterministic. It is also possible to have dynamic histogram plots for the three variables but this is left as an exercise for the reader.

We will now look at the second MCMC estimation method: Metropolis Hastings sampling.

1.9 Macro to run a hybrid Metropolis and Gibbs sampling method for a linear regression example

Our linear regression model has three unknown parameters and we have in the above macro updated all three using Gibbs sampling from the full conditional posterior distributions. We will now look at how we can replace the updating steps for the two fixed parameters, β_0 and β_1 with Metropolis steps.

We first need to load up the Metropolis macro file:

- Select **Open Macro** from the **File** Menu.
- Select **mhlr.txt** from the list of possible macros.

We will here discuss the step to update β_0 as the step for β_1 is similar. At each iteration, t, we firstly need to generate a new proposed value for β_0, β_0^*, and this is done in the macro by the following command:

```
▶ calc b30 = b6+b32*b21
```

Here b30 stores the new value (β_0^*), b6 is the current value ($\beta_0(t-1)$), b32 is the proposal distribution standard deviation and b21 is a random Normal(0,1) draw.

Next we need to evaluate the posterior ratio. It is generally easier to work with log-posteriors than posteriors so in reality we work with the log-posterior difference, which at step t is:

$$\begin{aligned} r_t &= p(\beta_0^*, \beta_1(t-1), \sigma_e^2(t-1)|y)/p(\beta_0(t-1), \beta_1(t-1), \sigma_e^2(t-1)|y) \\ &= \exp(\log(p(\beta_0^*, \beta_1(t-1), \sigma_e^2(t-1)|y)) \\ &\quad - \log(p(\beta_0(t-1), \beta_1(t-1), \sigma_e^2(t-1)|y))) \\ &= \exp(d_t) \end{aligned}$$

We then have

$$\begin{aligned} d_t = -\frac{1}{2\sigma_e^2(t-1)} \cdot \Bigg(&\sum_i (y_i - \beta_0^* - x_i\beta_1(t-1))^2 \\ &- \sum_i (y_i - \beta_0(t-1) - x_i\beta_1(t-1))^2 \Bigg) \end{aligned}$$

which with expansion and cancellation of terms can be written as

$$d_t = -\frac{1}{2\sigma_e^2(t-1)} \cdot \left(2 \left(\sum_i y_i - \beta_1(t-1) \sum_i x_i \right) \right.$$
$$\left. \cdot \left(\beta_0(t-1) - \beta_0^* + N((\beta_0^*)^2 - \beta_0^2(t-1)) \right) \right)$$

We evaluate this in the macro with the command

```
▶ calc b34 = -1*(2*(b7-b31)*(b15-b6*b12) + b13*(b31*b31 -
  b7*b7))/(2*b8)
```

Then to decide whether to accept or not, we need to compare a random uniform with the minimum of $(1, \exp(d_t))$. Note that if $d_t > 0$ then $\exp(d_t) > 1$ and so we always accept such proposals and in the macro we then only evaluate $\exp(d_t)$ if $d_t > 0$. This is important because as d_t becomes larger, $\exp(d_t) \rightarrow \infty$ and so if we try and evaluate it we will get an error. The accept/reject decision is performed via a **SWITch** command as follows in the macro:

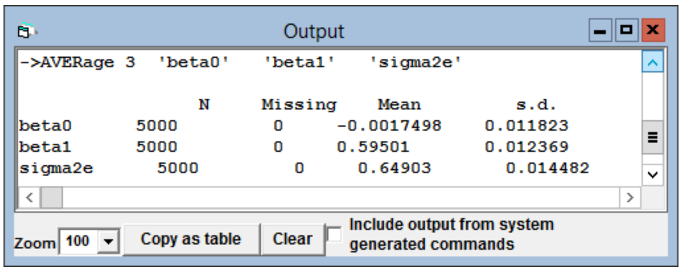
```
calc b35 = (b34 > 0)
switch b35
case 1 :
note definitely accept as higher likelihood
  calc b6 = b30
  calc b40 = b40+1
  leave
case 0 :
note only sometimes accept and add 1 to b40 if accept
  pick b1 c30 b36
  calc b6 = b6 + (b30-b6)*(b36 < expo(b34))
  calc b40 = b40 + 1*(b36 < expo(b34))
  leave
ends
```

Here b40 is storing the number of accepted proposals. As the macro language does not have an if statement the `calc b6 = b6 + (b30-b6)*(b36 < expo(b34))` statement is equivalent to an if that keeps b6 (β_0) at its current value if the proposal is rejected and sets it to the proposed value (b30) if it is accepted.

The step for β_1 has been modified in a similar manner. Here the log posterior ratio at iteration t after expansion and cancellation of terms becomes

$$d_t = -\frac{1}{2\sigma_e^2(t-1)} \cdot \left(2\left(\sum_i x_i y_i - \beta_0(t)\sum_i x_i\right) \cdot \left(\beta_1(t-1) - \beta_1^*\right) + \left((\beta_1^*)^2 - \beta_1^2(t-1)\right) \cdot \sum_i x_i^2\right)$$

To run this second macro we simply press the **Execute** button on the macro window. Again after some time the pointer will have changed back from the egg timer and the model will have run. As with the Gibbs sampling macro earlier we can now look at the estimates that are stored in c14–c16 via the **Averages and Correlations** window. This time we get the following:



```
->AVERage 3  'beta0'   'beta1'   'sigma2e'

                N      Missing     Mean         s.d.
beta0       5000          0     -0.0017498    0.011823
beta1       5000          0      0.59501      0.012369
sigma2e        5000          0      0.64903      0.014482
```

The difference in the estimates between the two macros is small and is due to the stochastic nature of the MCMC methods. The number of accepted proposals for both β_0 and β_1 is stored in boxes b40 and b41 respectively and so to work out the acceptance rates we can use the command interface window:

- Select **Command Interface** from the **Data Manipulation** menu.
- Type the following commands:

```
▶ Calc b40=b40/5500
▶ Calc b41=b41/5500
```

These commands will give the following acceptance rates:

```
 ->calc b40=b40/5500
0.75655
->calc b41=b41/5500
0.74291
```

So we can see that both parameters are being accepted about 75% of the time. The acceptance rate is inversely related to the proposal distribution variance and one of the difficulties in using Metropolis Hastings algorithms is choosing a suitable value for the proposal variance. There are situations to avoid at both ends of the proposal distribution scale. Firstly choosing too large a proposal variance will mean that proposals are rarely accepted and this will induce a highly autocorrelated chain. Secondly choosing too small a proposal variance will mean that although we have a high acceptance rate the moves proposed are small and so it takes many iterations to explore the whole parameter space again inducing a highly autocorrelated chain. In the example here, due to the centering of the predictor we have very little correlation between our parameters and so the high (75%) acceptance rate is OK. Generally however we will aim for lower acceptance rates.

To investigate this further the interested reader might try altering the proposal distribution standard deviations (the lines `calc b32 = 0.01` and `calc b33 = 0.01` in the macro) and seeing the effect on the acceptance rate. It is also interesting to look at the effect of using MH sampling via the parameter traces described earlier.

1.10 MCMC estimation of multilevel models in MLwiN

The linear regression model we have considered in the above example can be fitted easily using least squares in any standard statistics package. The MLwiN macro language that we have used to fit the above model is a compiled language and is therefore computationally fairly slow. In fact the speed difference will become evident when we fit the same model with the MLwiN MCMC engine in the next chapter. If users wish, to improve their understanding of MCMC, they can write their own macro code for fitting more complex models in MCMC and the algorithms for many basic multilevel models are given in Browne (1998). Their results could then be compared with those obtained using the MCMC engine.

The MCMC engine can be used to fit many multilevel models and many extensions. As was described earlier, MCMC algorithms involve splitting the unknown parameters into blocks and updating each block in a separate step. This means that extensions to the standard multilevel models generally involve simply adding extra steps to the algorithm. These extra steps will be described when these models are introduced.

In the standard normal models that are the focus of the next few chapters we use Gibbs sampling for all steps although the software allows the option to change to univariate Metropolis sampling for the fixed effects and residuals. The parameters are blocked in a two level model into the fixed effects, the level 2 random effects (residuals), the level 2 variance matrix and the level 1 variance. We then update the fixed effects as a block using a multivariate normal draw from the full conditional, the level 2 random effects are updated in blocks, 1 for each level 2 unit again by multivariate normal draws. The level 2 variance matrix is updated by drawing from its inverse-Wishart full conditional and the level 1 variance from its inverse Gamma full conditional. For models with extra levels we have additional steps for the extra random effects and variance matrix.

Chapter learning outcomes

- ⋆ Some theory behind the MCMC methods
- ⋆ How to calculate full conditional distributions
- ⋆ How to write MLwiN macros to run the MCMC methods
- ⋆ How MLwiN performs MCMC estimation.

Chapter 2

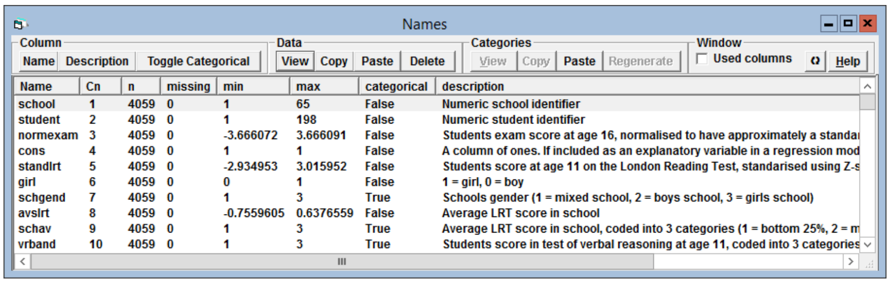
Single Level Normal Response Modelling

In this chapter we will consider fitting simple linear regression models and normal general linear models. This will have three main aims: to start the new user off with models they are familiar with before extending our modelling to multiple levels; to show how such models can be fitted in MLwiN, and finally to show how these models can be fit in a Bayesian framework and to introduce a model comparison diagnostic DIC (Spiegelhalter et al., 2002) that we will also be using in the models in later chapters.

We will consider here an examination dataset stored in the worksheet **tutorial.ws**. This dataset will be used in many of the chapters in this manual and is also the main example dataset in the MLwiN user's guide (Rasbash et al., 2008). To view the variables in the dataset you need to load up the worksheet as follows:

- Select **Open Sample Worksheet** from the **File** menu.
- Select **tutorial.ws**.

This will open the following **Names** window:



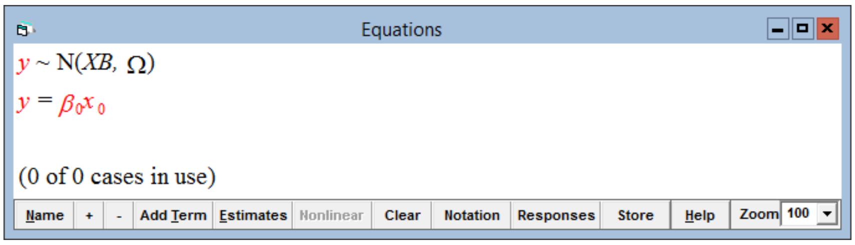
Name	Cn	n	missing	min	max	categorical	description
school	1	4059	0	1	65	False	Numeric school identifier
student	2	4059	0	1	198	False	Numeric student identifier
normexam	3	4059	0	-3.666072	3.666091	False	Students exam score at age 16, normalised to have approximately a standa
cons	4	4059	0	1	1	False	A column of ones. If included as an explanatory variable in a regression mod
standlrt	5	4059	0	-2.934953	3.015952	False	Students score at age 11 on the London Reading Test, standarised using Z-s
girl	6	4059	0	0	1	False	1 = girl, 0 = boy
schgend	7	4059	0	1	3	True	Schools gender (1 = mixed school, 2 = boys school, 3 = girls school)
avslrt	8	4059	0	-0.7559605	0.6376559	False	Average LRT score in school
schav	9	4059	0	1	3	True	Average LRT score in school, coded into 3 categories (1 = bottom 25%, 2 = m
vrband	10	4059	0	1	3	True	Students score in test of verbal reasoning at age 11, coded into 3 categories

Our response of interest is named **normexam** and is a (normalised) total exam score at age 16 for each of the 4059 students in the dataset. Our main predictor of interest is named **standlrt** and is the (standardised) marks achieved in the London reading test (LRT) taken by each student at age 11. We are interested in the predictive strength of this variable and we can measure this by looking at how much of the variability in the exam score is explained by a simple linear regression on LRT. Note that this is the model we fitted using macros in the last chapter.

We will set up the linear regression via MLwiN's Equations window that can be accessed as follows:

- Select **Equations** from the **Model** menu.

The **Equations** window will then appear:


Equations

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

$y = \beta_0 x_0$

(0 of 0 cases in use)

How to set up models in MLwiN is explained in detail in the User's Guide to MLwiN and so we will simply reiterate the procedure here but generally less detail is given in this manual.

We now have to tell the program the structure of our model and which columns hold the data for our response and predictor variables. We will firstly define our response (y) variable to do this:

- Click on **y** (either of the **y** symbols shown will do).
- In the **y** list, select **normexam**.

We will next set up the structure of the model. We will be extending the model to 2 levels later, so for now we will specify two levels although the model itself will be 1 level. The model is set up as follows:

- In the **N levels** list, select **2-ij**.
- In the **level 2(j):** list, select **school**.
- In the **level 1(i):** list, select **student**.
- Click on the **done** button.

In the **Equations** window the red y has changed to a black y_{ij} to indicate that the response and the first and second level indicators have been defined. We now need to set up the predictors for the linear regression model:

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

Note that **cons** is a column containing the value 1 for every student and will hence be used for the intercept term. The **fixed parameter** tick box is checked by default and so we have added to our model a fixed intercept term. We also need to set up residuals so that the two sides of the equation balance. To do this:

- Check the box labelled **i(student)**.
- Click on the **Done** button.

Note that we specify residuals at the student level only as we are fitting a single-level model. We have now set up our intercept and residuals terms but to produce the linear regression model we also need to include the slope (**standlrt**) term. To do this we need to add a term to our model as follows:

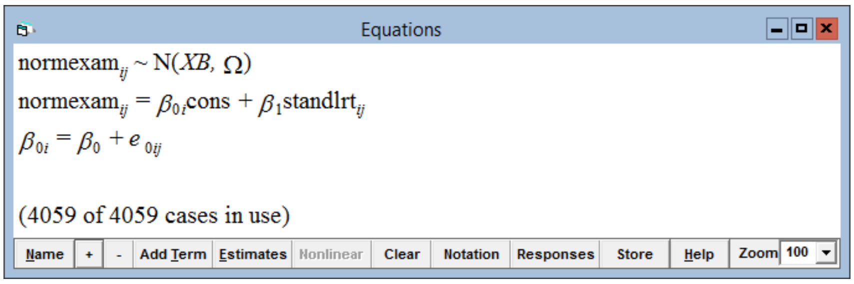
- Click the **Add Term** button on the tool bar.
- Select **standlrt** from the variable list.
- Click on the **Done** button.

Note that this adds a fixed effect only for the **standlrt** variable. Until we deal with complex variation in a later chapter we will ALWAYS only have one set of residuals at level 1, i.e. only one variable with the level 1 tick box checked.

We have now added all terms for the linear regression model and if we look at the **Equations** window and:

- Click the + button on the tool bar to expand the model definition

we get:



$\text{normexam}_{ij} \sim \text{N}(XB, \Omega)$

$\text{normexam}_{ij} = \beta_{0i}\text{cons} + \beta_1\text{standlrt}_{ij}$

$\beta_{0i} = \beta_0 + e_{0ij}$

(4059 of 4059 cases in use)

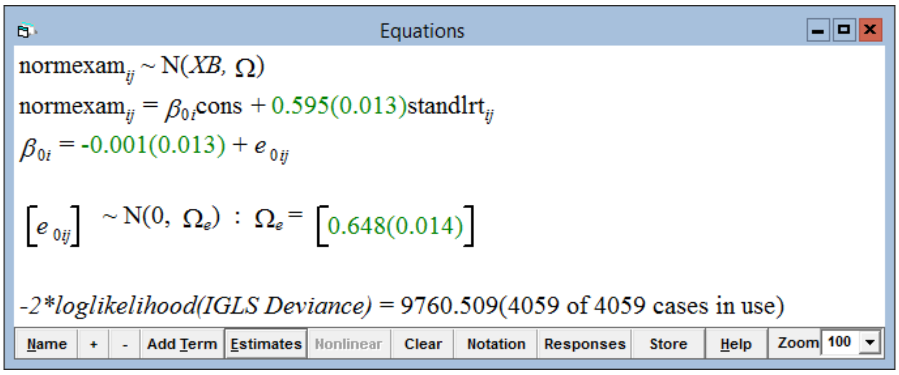
If we substitute the third line of the model into the second line and remember that **cons** $= 1$ for all students we get $y_{ij} = \beta_0 + \beta_1 \textbf{standlrt}_{ij} + e_{ij}$, the standard linear regression formula. To fit this model we now simply:

- Click **Start**.

This will run the model using the default iterative generalised least squares (IGLS) method. You will see that the model only takes one iteration to converge and this is because for a 1 level model the IGLS algorithm is equivalent to ordinary least squares and the estimates produced should be identical to the answer given by any standard statistics package regression routine. To get the numerical estimates:

- Click twice on the **Estimates** button.

This will produce the following screen:



$\text{normexam}_{ij} \sim \text{N}(XB, \Omega)$

$\text{normexam}_{ij} = \beta_{0i}\text{cons} + 0.595(0.013)\text{standlrt}_{ij}$

$\beta_{0i} = -0.001(0.013) + e_{0ij}$

$\left[e_{0ij}\right] \sim \text{N}(0, \Omega_e) \ : \ \Omega_e = \left[0.648(0.014)\right]$

$-2*loglikelihood(IGLS\ Deviance)$ = 9760.509(4059 of 4059 cases in use)

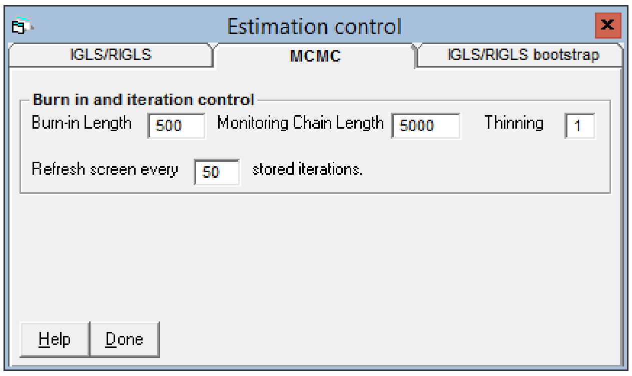
Here we see that there is a positive relationship between exam score and LRT score (slope coefficient of 0.595). Our response and LRT scores have been normalised i.e. they have mean 0 and variance 1, and so the LRT scores explain $(1 - 0.648) \times 100 = 35.2\%$ of the variability in the response variable.

As this manual is generally about the MCMC estimation methods in MLwiN we will now fit this model using MCMC. Note that it is always necessary in MLwiN to run the IGLS or RIGLS estimation methods prior to running MCMC as these methods set up the model and starting values for the MCMC methods.

To run MCMC:

- Click on the **Estimation Control** button
- Select the tab labelled **MCMC**

The window should then look as follows:


Estimation control

IGLS/RIGLS | MCMC | IGLS/RIGLS bootstrap

Burn in and iteration control

Burn-in Length 500 Monitoring Chain Length 5000 Thinning 1

Refresh screen every 50 stored iterations.

As described in the previous chapter MLwiN uses a mixture of Gibbs sampling steps when the full conditionals have simple forms, and Metropolis Hastings steps when this is not the case. Here the estimation control window shows the default settings for burn-in length, run length, thinning and refresh rate. All other MCMC settings are available from the **Advanced MCMC Methodology Options** window available from the MCMC submenu of the Model menu.

In this release 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 Methodology Options** window 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.

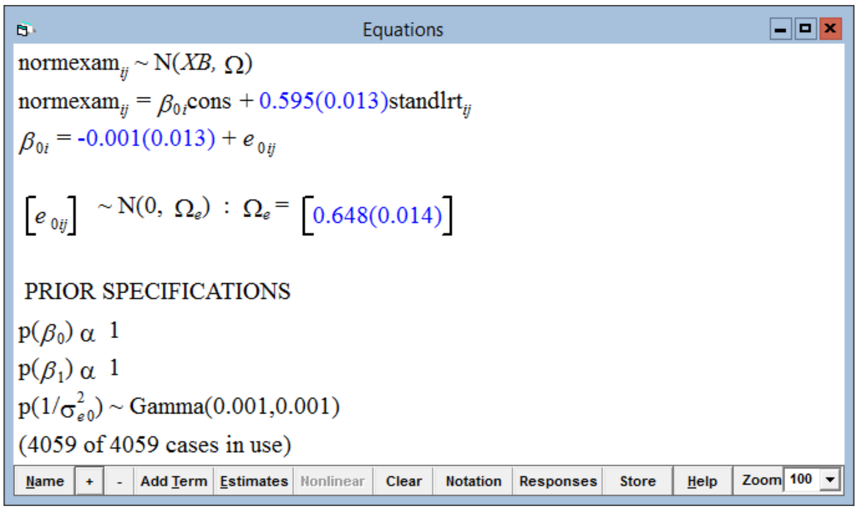
Thinning. This is the frequency with which successive values in the Markov chain are stored. This works in a more intuitive way in this release, for example running a chain for a monitoring chain length of 50,000 and setting thinning to 10 will result in 5,000 values being stored. The default value of 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.

For our simple linear regression model we will simply use the default settings. With regards to prior distributions we will also use the default priors as described in the last chapter. In this release for clarity the prior distributions are included in the Equations window. They can be viewed by:

- Clicking on the + button on the toolbar.

This will then give the following display (note the estimates are still the IGLS estimates as we have not yet started the MCMC method.)



$\text{normexam}_{ij} \sim \text{N}(XB, \Omega)$

$\text{normexam}_{ij} = \beta_{0i}\text{cons} + 0.595(0.013)\text{standlrt}_{ij}$

$\beta_{0i} = -0.001(0.013) + e_{0ij}$

$\left[e_{0ij}\right] \sim \text{N}(0, \Omega_e) : \Omega_e = \left[0.648(0.014)\right]$

PRIOR SPECIFICATIONS

$\text{p}(\beta_0) \propto 1$

$\text{p}(\beta_1) \propto 1$

$\text{p}(1/\sigma^2_{e0}) \sim \text{Gamma}(0.001,0.001)$

(4059 of 4059 cases in use)

2.1 Running the Gibbs Sampler

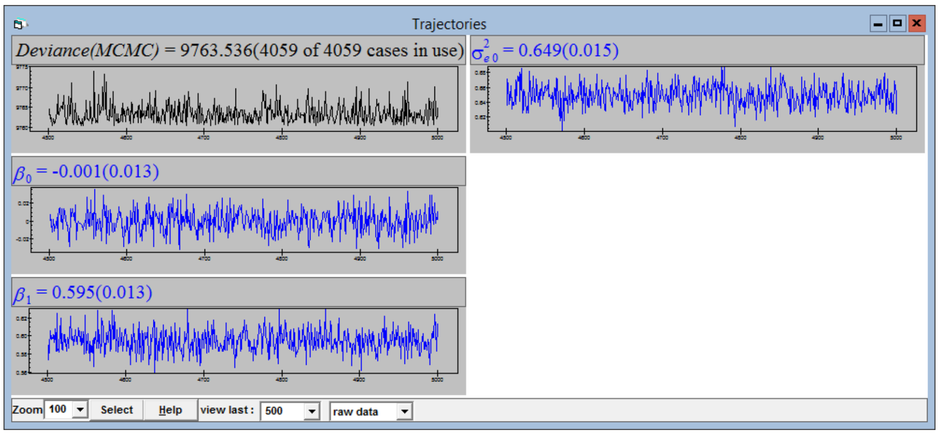
We will now run the simple linear regression model using MCMC. Before we start we will also open the **Trajectories** window so that we can see the dynamic chains of parameter estimates as the method proceeds (note that although viewing the chains is useful the extra graphical overhead means that the method will run slower).

- Select **Trajectories** from the **Model** menu.

It is best to reposition the two windows so that both the equations and chains are visible then we start estimation by:

- Clicking the **Start** button.

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:


Trajectories
Deviance(MCMC) = 9763.536(4059 of 4059 cases in use)
σ^2_{e0} = 0.649(0.015)
β_0 = -0.001(0.013)
β_1 = 0.595(0.013)
Zoom 100
Select
Help
view last : 500
raw data

These graphs show the estimates for each of the three parameters in our model and the deviance statistic for each of the last 500 iterations. The numbers given in both the Equations window and the Trajectories window are the mean estimates for each of the parameters (including the deviance) based on the run of 5,000 iterations (with the standard deviation of these 5,000 estimates given in brackets). It should be noted that in this example we have almost identical estimates as the least squares estimates which given we have used ‘diffuse’ priors is reassuring.

Healthy Gibbs sampling traces should look like any of these iteration traces; 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 we are showing 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. It is possible to get more detailed information about each parameter and to assess whether we have run our chains for long enough. For now we will assume we have run for long enough and consider MCMC diagnostics in the next chapter.

2.2 Deviance statistic and the DIC diagnostic

The deviance statistic (McCullagh and Nelder, 1989) can be thought of as a measure of how well our model fits the data. Generally the deviance is the difference in $-2\times$log(likelihood) values for the fitted model and a saturated model. In the normal model case we have:

$$\log(\text{likelihood}) = -\frac{N}{2}\log(2\pi\hat{\sigma}_e^2) - \frac{1}{2\hat{\sigma}_e^2}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

where N is the number of lowest level units (students) in the dataset, $\hat{\sigma}_e^2$ is an estimate of the level 1 variance and $\hat{y}_i$ is the predicted value for student i, in the case of the linear regression model $\hat{y}_i = \hat{\beta}_0 + X_i\hat{\beta}_1$. For the saturated model we have $y_i = \hat{y}_i \forall i$ and so the second term in the log-likelihood equals zero. In the diagnostic that follows we are interested in differences in the deviance and so we will assume the deviance of the saturated model is zero as this term will cancel out.

Spiegelhalter et al. (2002) use the deviance with MCMC sampling to derive a diagnostic known as the Deviance Information Criterion (DIC), which is a generalization of the Akaike's Information Criterion (AIC - See MLwiN help system for more details). The DIC diagnostic is simple to calculate from an MCMC run as it simply involves calculating the value of the deviance at each iteration, and the deviance at the expected value of the unknown parameters ($D(\bar{\theta})$). Then we can calculate the 'effective' number of parameters (p_D) by subtracting $D(\bar{\theta})$ from the average deviance from the 5000 iterations ($\bar{D}$). The DIC diagnostic can then be used to compare models as it consists of the sum of two terms that measure the 'fit' and the 'complexity' of a particular model,

$$DIC = \bar{D} + p_D = D(\bar{\theta}) + 2p_D = 2\bar{D} - D(\bar{\theta}).$$

It should be noted that the DIC diagnostic has not had universal approval and the interested reader should read the discussion of the paper. Note that in normal response models we have the additional parameter $\hat{\sigma}_e^2$. In calculating $D(\bar{\theta})$we use the arithmetic mean of $\hat{\sigma}_e^2$, $(E(\sigma_e^2))$ as this generalizes easily to multivariate normal problems.

To calculate the DIC diagnostic for our model:

- Select **MCMC/DIC diagnostic** from the **Model** menu.

This will bring up the **Output** window with the following information:

Dbar	D(thetabar)	pD	DIC
9763.54	9760.51	3.02	9766.56

Note that the value 9760.51 for $D(\bar{\theta})$ is (almost) identical to the $-2\times$log-likelihood value given for the IGLS method for the same model. For 1 level models this will always be true but when we consider multilevel models this will no longer be true. Also in this case the effective number of parameters is (approximately) the actual number of parameters in this model. When we consider fitting multilevel models this will again no longer be the case.

2.3 Adding more predictors

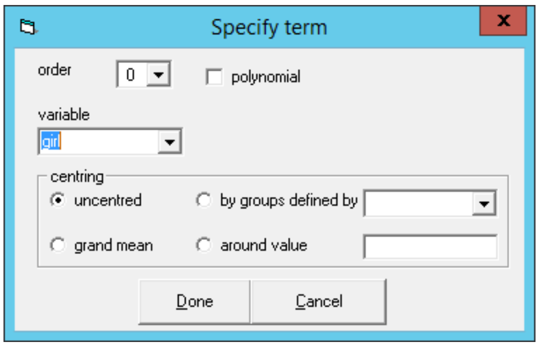
In this dataset we have two more predictors we will investigate, gender and school gender. Both of these variables are categorical and we can use the **Add term** button to create dummy variable fixed effects, which are then added to our model. To set up these new parameters we MUST change estimation mode back to IGLS/RIGLS before altering the model.

- Click on the **IGLS/RIGLS** tab on the **Estimation control** window.

We now wish to set up a model that includes an effect of gender (**girl**) and two effects for the school types (**boysch** and **girlsch**) with the base class for our model being a boy in a mixed school. To set this up in the main effects and interactions window we need to do the following:

- Click on the **Add Term** button on the **Equations** window.
- Select **girl** from the **variable** pull-down list.

The **Specify term** window should look as follows:



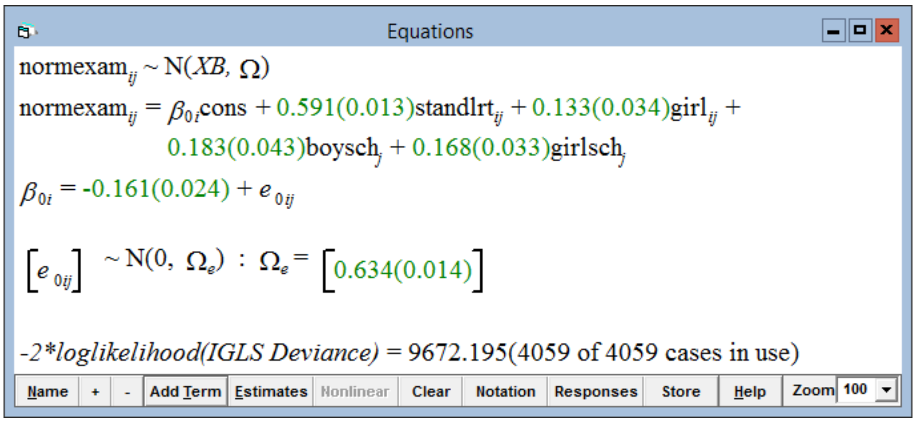
Now if we click on the **Done** button a term named **girl** will be added to the model. We need now to additionally add school gender effects:

- Click on the **Add Term** button on the **Equations** window.
- Select **schgend** from the **variable** pull down list.
- Click on the **Done** button.

Having successfully performed this operation we will run the model using IGLS.

- Click on the **Start** button.

This will then give the following in the **Equations** window:



$\text{normexam}_{ij} \sim \text{N}(XB, \Omega)$

$\text{normexam}_{ij} = \beta_{0i}\text{cons} + 0.591(0.013)\text{standlrt}_{ij} + 0.133(0.034)\text{girl}_{ij} + 0.183(0.043)\text{boysch}_j + 0.168(0.033)\text{girlsch}_j$

$\beta_{0i} = -0.161(0.024) + e_{0ij}$

$\begin{bmatrix} e_{0ij} \end{bmatrix} \sim \text{N}(0, \Omega_e) : \Omega_e = \begin{bmatrix} 0.634(0.014) \end{bmatrix}$

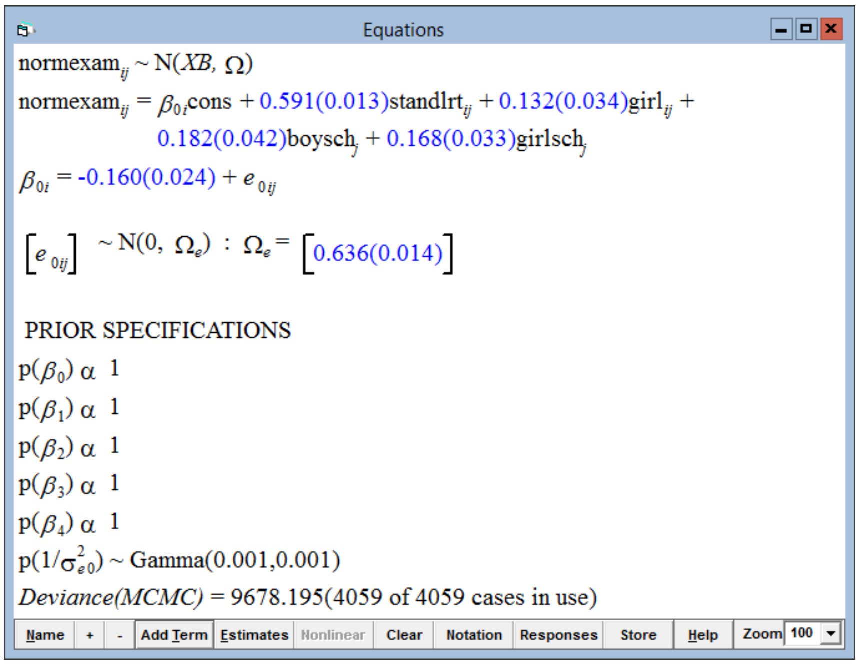
*-2*loglikelihood(IGLS Deviance)* = 9672.195(4059 of 4059 cases in use)

So we see (by comparing the fixed effects estimates to their standard errors) that in mixed schools, girls do significantly better than boys and that students in single sex schools do significantly better than students of the same sex in mixed schools. These additional effects have explained only a small amount of the remaining variation, the residual variance has reduced from 0.648 to 0.634.

To fit this model using MCMC:

- Click on the **MCMC** tab on the **Estimation Control** window.
- Click **Done**.
- Click on the **Start** button.

After running for 5000 iterations we get the following estimates:



$\text{normexam}_{ij} \sim \text{N}(XB, \Omega)$

$\text{normexam}_{ij} = \beta_{0i}\text{cons} + 0.591(0.013)\text{standlrt}_{ij} + 0.132(0.034)\text{girl}_{ij} + 0.182(0.042)\text{boysch}_j + 0.168(0.033)\text{girlsch}_j$

$\beta_{0i} = -0.160(0.024) + e_{0ij}$

$\left[e_{0ij}\right] \sim \text{N}(0, \Omega_e) : \Omega_e = \left[0.636(0.014)\right]$

PRIOR SPECIFICATIONS

$p(\beta_0) \alpha\ 1$

$p(\beta_1) \alpha\ 1$

$p(\beta_2) \alpha\ 1$

$p(\beta_3) \alpha\ 1$

$p(\beta_4) \alpha\ 1$

$p(1/\sigma^2_{e0}) \sim \text{Gamma}(0.001,0.001)$

Deviance(MCMC) = 9678.195(4059 of 4059 cases in use)

Here again MCMC gives (approximately) the same estimates as least squares and if we now wish to compare our new model with the last model we can again look at the DIC diagnostic:

- Select **MCMC/DIC** diagnostic from the **Model** menu.

If we compare the output from the two models we have:

Dbar	D(thetabar)	pD	DIC
9763.54	9760.51	3.02	9766.56
9678.19	9672.21	5.99	9684.18

so that adding the 3 parameters has increased the effective number of parameters to 6 (5.99) but the deviance has been reduced by approximately 88 meaning that the DIC has reduced by around 82 and so the DIC diagnostic suggests this is a better model. Note that the DIC diagnostic accounts for the number of parameters in the two models and so the two DIC values are directly comparable and so any decrease in DIC suggests a better model. However due to the stochastic nature of the MCMC algorithm there will be random variability in the DIC diagnostic depending on starting values and random number seeds and so if a model gives only a small difference in DIC you should confirm if this is a real difference by checking the results with different seeds and/or starting values.

2.4 Fitting school effects as fixed parameters

We have in the last model seen that whether a school is single sex or mixed has an effect on its pupils' exam scores. We can now take this one step further (as motivation for the multilevel models that follow) by considering fitting a fixed effect for each school in our model. To do this we will first have to set up the **school** variable as categorical:

- Select **Names** from the **Data Manipulation** menu.
- Note that the **school** variable is highlighted.
- Click on the **Toggle Categorical** button on the **Names** window.
- Click on the **Categories** button.
- Click on the **OK** button on the window that appears.

This will set up school names coded **school_1** to **school_65** for schools 1 to 65 which will be OK for our purposes, however generally we could have input all the categories for example school names here.

We will now use the **Add Term** button to set up the school effects. We will for now replace the school gender effects as they will be confounded with the school effects. Note again that as we are about to modify the model structure we will need to:

- Change estimation mode to IGLS/RIGLS via the **Estimation Control** window.

Next we set up the fixed effects as follows:

- Click **Estimates** in the **Equations** window once.
- Click on the β_4 (**girlsch**) term.
- Click on the **Delete Term** button and respond **Yes** to removing all **schgend** terms.
- Select the **Add Term** button from the **Equations** window.
- Select **school** from the variable list and click on the **Done** button.

This will now have removed the **schgend** terms from the model and set up 64 dummy variables for the **school** fixed effects using **school_1** as a base category. You will notice that all the school fixed effects have now been added to the model in the **Equations** window:

- Click the **Start** button.

This will run the model using least squares (in 1 iteration) and give estimates for the 64 school effects. Note that these effects can be thought of as differences in average achievement for the 64 schools when compared to the base school. To fit this model in MCMC we need to:

- Select **MCMC** from the **Estimation** menu.
- Click on the **Start** button.

This model has 67 fixed effects and so even with the block updating Gibbs sampling algorithm it will take a few minutes to run for 5000 iterations. After running we see that the estimate for the base school (β_0) is 0.341 (0.090) so that this school is significantly better than average (for a boy with average **standlrt** mark) and all other school effect estimates ($\beta_3,\ldots,\beta_{66}$) are relative to this school.

If we were to check the DIC diagnostic for this model we have:

Dbar	D(thetabar)	pD	DIC
9183.46	9115.36	68.10	9251.56

The DIC value has reduced from 9684 in our last model to 9252, a reduction of 432 points showing that the school in which the student studies has a strong effect on their exam mark. Note that the effective number of parameters, 68.10, is still approximately correct for this model.

The variance estimate σ^2_{e0} has now been reduced to 0.563 and so we have now explained 43.7% of the variation in our original response with the addition of 67 fixed effects.

In this example we have introduced fixed school effects and shown that we actually do not need to fit a random effects model to account for school differences. We will however in the next chapter introduce multilevel modelling by fitting school effects as random terms and explain why and when this may be a better approach.

Chapter learning outcomes

- ⋆ How to set up models in MLwiN using the Equations window.
- ⋆ How to set up 1 level models in MLwiN.
- ⋆ How to run the MCMC Gibbs sampling method.
- ⋆ How to access and interpret the DIC diagnostic.
- ⋆ How to fit a fixed effects model.

Chapter 3

Variance Components Models

We ended the last chapter with an example of how to fit school effects as fixed terms in a linear model. In this chapter we will introduce fitting these same school effects as random terms. Whether you choose to fit terms as fixed or random is one of the main difficulties faced by researchers new to multilevel modelling. In some scenarios the answer is obvious but in other situations which model you fit will very much depend on what your main research questions are and the context in which the data are collected.

Here we consider how to add a categorical explanatory variable to our model. Certain categorical variables, for example gender and school gender in the tutorial example, will ALWAYS be fitted as fixed effects. This is because these variables have a limited number of categories and all categories are present in the dataset. The motivation behind fitting a categorical variable, for example school, as a set of random effects is that the categories of this variable that we observe are, in fact, a sample of the possible categories. Then, just like our observations at level 1, we can think of our categories as being a sample from a population of categories and make a distributional assumption about these categories.

The main reason for fitting a categorical variable as a random term rather than as fixed effects is if our primary interest is in the variability across the various categories rather than inferences about any single category. For example we may want to calculate how much of the variability in our outcome variable is due to the schools attended and how much is residual variation due to pupil differences. Also as we may only have a small sample of level 1 units for each category, the random effects produced will be more conservative than the category effects produced by a fixed effect model. This is because we use the fact that categories, for examples schools, are similar to each other, in that case we may borrow strength from the other schools and when we wish to estimate them, we “shrink” the school effects towards the average school effect.

In multilevel modelling when we treat a categorical variable as a set of random effects we describe the set of categories as a level in the model. This is because our observations (level 1 in the model) are nested within these categories, for example pupils are nested within schools. The level terminology can be extended for example to 3 levels if we want to extend our model to contain both effects for schools, and for the local education authorities (LEAs) to which the schools belong. Here we have a nesting of pupils within schools and schools within LEAs and hence a 3 level structure. Note that we will see in later chapters that structures are not always nested, leading to cross-classified structures. Here we will use the alternative terminology of classification rather than level.

Levels are not the same as random effects as there may be several sets of random effects at a level; such models, called random slopes regression models, are described in a later chapter. Having more than one set of random effects in a model can be thought of as the random equivalent of having an interaction between a categorical variable and another explanatory variable in the model as we will see in the later chapter.

For now, to distinguish between levels and random effects, we will have *school* as the level and the *school intercepts* as the random effects.

3.1 A 2 level variance components model for the Tutorial dataset

We will now return our attention to the tutorial dataset. At the end of the last chapter we had fitted the school fixed effects model. This time we will fit a school random effects model. To do this we will have to remove all the fixed effects that are currently in the model, This may be done by reloading the worksheet **tutorial.ws** or:

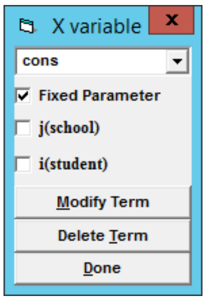
- In the **Equations** window click on the **Clear** button.

This will reset our model and we will have to set up our model from scratch. Now we need to set up the linear regression model that contains the intercept and the **standlrt** explanatory variable, which was fitted first in the last chapter. (If you are unsure how to do this follow the instructions in the last chapter.)

We now have to add random school level intercepts to the model. Note that to do this you should have the estimation method set to IGLS. The variable **cons** is associated with the intercepts and so you need to do the following:

- Click on **cons**

and the following X variable screen will appear:



We now need to:

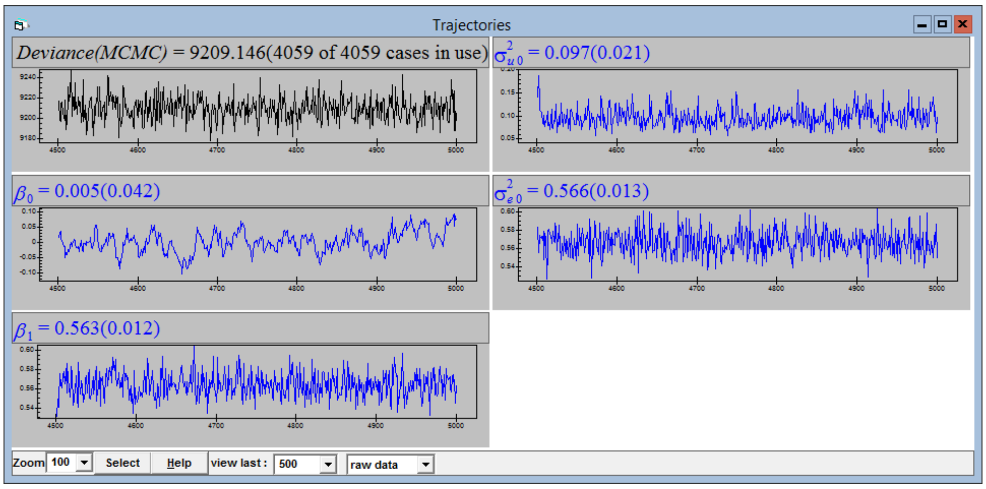
- Click on the **i(student)** box
- Click on the **j(school)** box
- Click on the **Done** button

to allow for level 1 variation and random intercepts for each school. Note that you may already have the **i(student)** box ticked from the regression in which case you need to ensure it is still ticked. The model we have now set up is a member of the **variance components** family. A model is called a variance components model if there is only one set of random effects (intercepts) for each level in the model. This is because the model splits the total variation into components of variation for each level in the model. The particular variance components model with an intercept and slope term in the fixed part is often called a **random intercepts** model. This is because graphically (as shown in the User's Guide to MLwiN), each school can be represented by a (parallel) regression line with a fixed slope and a random intercept.

We will now run the model firstly using IGLS to obtain starting values, and then using MCMC with the default settings:

- Click on the **Start** button.
- Select **MCMC** from the **Estimation** menu.
- Click on the **Start** button.

Again if you have the **Equations** and **Trajectories** windows open you will see the estimates and traces change as the estimation proceeds. Upon completion of the 5,000 iterations the **Trajectories** window should look as follows:



Deviance(MCMC) = 9209.146(4059 of 4059 cases in use)

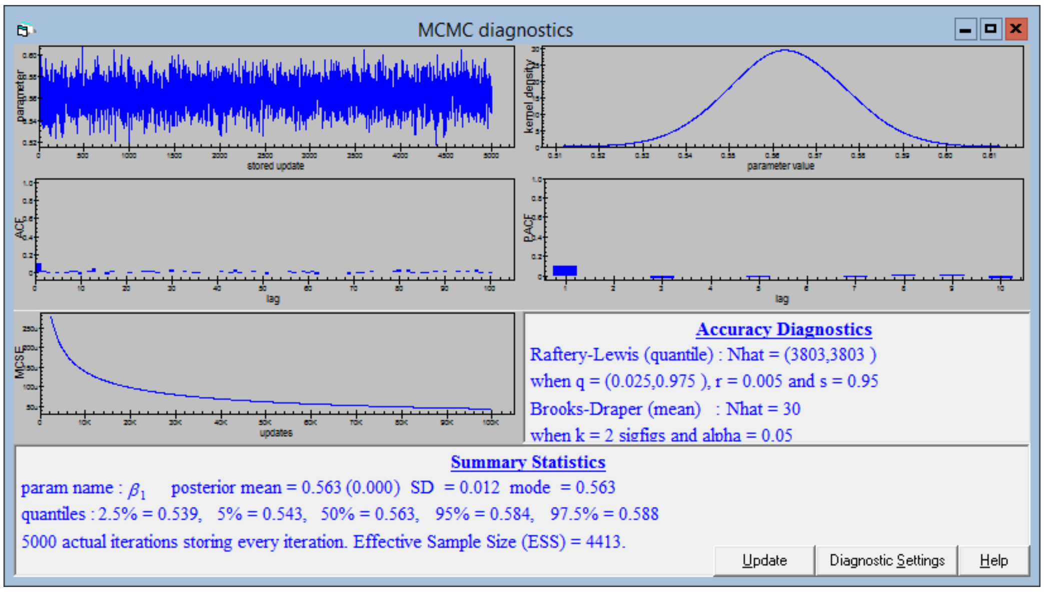
σ^2_{u0} = 0.097(0.021)

β_0 = 0.005(0.042)

σ^2_{e0} = 0.566(0.013)

β_1 = 0.563(0.012)

Here we see that, unlike the linear regression example in the last chapter, these traces do not all look healthy and the trace for β_0 looks quite autocorrelated i.e. each value of the trace is highly correlated with the preceding value. We can get more detailed diagnostic information about a parameter, for example the slope coefficient β_1, by clicking 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:



Accuracy Diagnostics

Raftery-Lewis (quantile) : Nhat = (3803,3803)
when q = (0.025,0.975), r = 0.005 and s = 0.95
Brooks-Draper (mean) : Nhat = 30
when k = 2 sigfigs and alpha = 0.05

Summary Statistics

param name : β_1 posterior mean = 0.563 (0.000) SD = 0.012 mode = 0.563
quantiles : 2.5% = 0.539, 5% = 0.543, 50% = 0.563, 95% = 0.584, 97.5% = 0.588
5000 actual iterations storing every iteration. Effective Sample Size (ESS) = 4413.

The upper left-hand cell simply reproduces the whole trace for the parameter. The upper right-hand cell gives a kernel density (which is like a smoothed histogram) estimate of the posterior distribution; when an *informative* prior distribution is used the density for this distribution is also displayed in black (see examples in later chapters). We can see in this example that the density looks to have approximately a Normal distribution. The second row of boxes plots 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. The MCSE is an indication of the accuracy of the mean estimate (MCSE = SD/$\sqrt{n}$, where SD is the standard deviation from the chain of values, and n is the number of iterations). This graph 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 & 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 defaults are 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 0.56 (2 significant figures) with the desired accuracy requires the chain to be run only for 30 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.

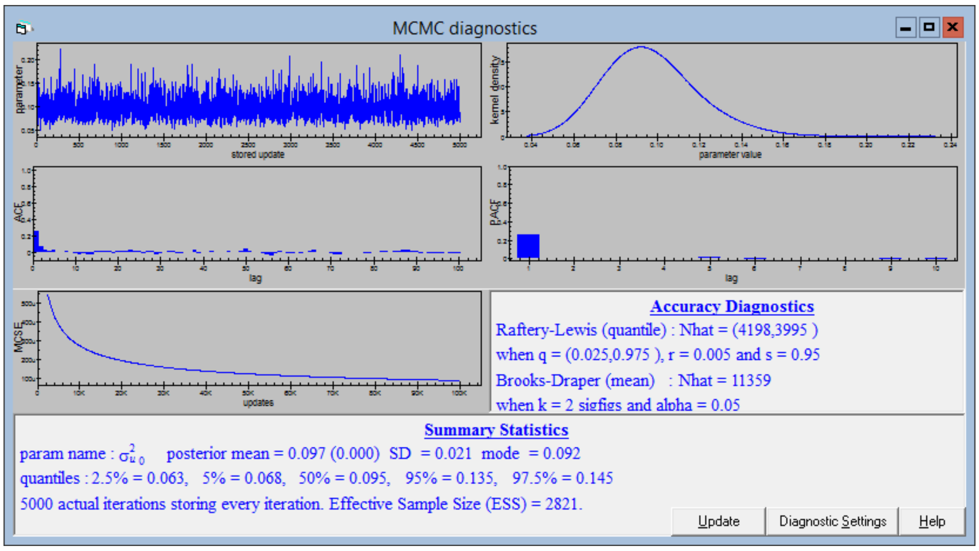
The bottom box contains some numerical summaries of the data. As well as the mean (with its MCSE in parenthesis), this box also contains the mode and median estimates. To estimate both 90% and 95% intervals this box also contains the appropriate quantiles of the distribution. For example a 95% central interval (Bayesian credible interval) runs from 0.539 to 0.588.

Also in the bottom row of the box details of the run length of the Markov chain are given. We also include an estimate of the effective (independent) sample size (see Kass et al., 1998). Here the number of stored iterations is divided by a measure of the correlation of the chain called the autocorrelation time κ where

$$\kappa = 1 + 2\sum_{k=1}^{\infty} \rho(k)$$

To approximate this value, we evaluate the sum up to $k = 5$ and then every subsequent value of k until $\rho(k) < 0.1$. So in this example we have an almost independent chain and our actual sample of 5,000 iterations is equivalent to an independent sample of 4,413 iterations.

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 (shown below).



Accuracy Diagnostics

Raftery-Lewis (quantile) : Nhat = (4198,3995)
when q = (0.025,0.975), r = 0.005 and s = 0.95
Brooks-Draper (mean) : Nhat = 11359
when k = 2 sigfigs and alpha = 0.05

Summary Statistics

param name : σ^2_{u0} posterior mean = 0.097 (0.000) SD = 0.021 mode = 0.092
quantiles : 2.5% = 0.063, 5% = 0.068, 50% = 0.095, 95% = 0.135, 97.5% = 0.145
5000 actual iterations storing every iteration. Effective Sample Size (ESS) = 2821.

Here we can see in the kernel density plot that the posterior distribution is not symmetric which is to be expected for a variance parameter. The Raftery-Lewis diagnostics suggest that we have run for long enough although to quote the mean estimate as 0.097 with 95% confidence the Brooks-Draper diagnostic suggests we run for 11,365 iterations. We see in the summary statistics that the 90% and 95% central credible interval that we can calculate from the quantiles will reflect the skewed nature of the posterior distribution. Also we see that the mode is less than the mean due to the long tail to the right of the distribution.

Finally, we can compare the results from Gibbs to the results from the IGLS method for this model in the following table:

	Gibbs Posterior		IGLS	
Parameter	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^2_{u0}	0.097	0.021	0.092	0.018
σ^2_{e0}	0.566	0.013	0.566	0.013

The only real difference is the slightly higher value for the Gibbs estimate of the level 2 variance. The Gibbs estimate for the mode of 0.092 (see above) is identical to the IGLS 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). We will talk about the choice of prior in more detail in a later chapter.

3.2 DIC and multilevel models

We can now work out the DIC diagnostic for this model via the **Model** menu:

- Select **MCMC/DIC diagnostic** from the **Model** menu.

Dbar	D(thetabar)	pD	DIC
9209.15	9146.16	59.98	9269.13

Here we will notice a difference between random effects models and fixed effects models. If we were to assume fixed school effects then our model would have 65 school intercepts plus the level 1 variance plus the fixed slope effect resulting in 67 parameters. Here however the school effects are related through the fact that they have a common variance structure and so we do not have independent 67 parameters. In fact, in this model the DIC diagnostic estimates the number of independent parameters to be approximately 60.

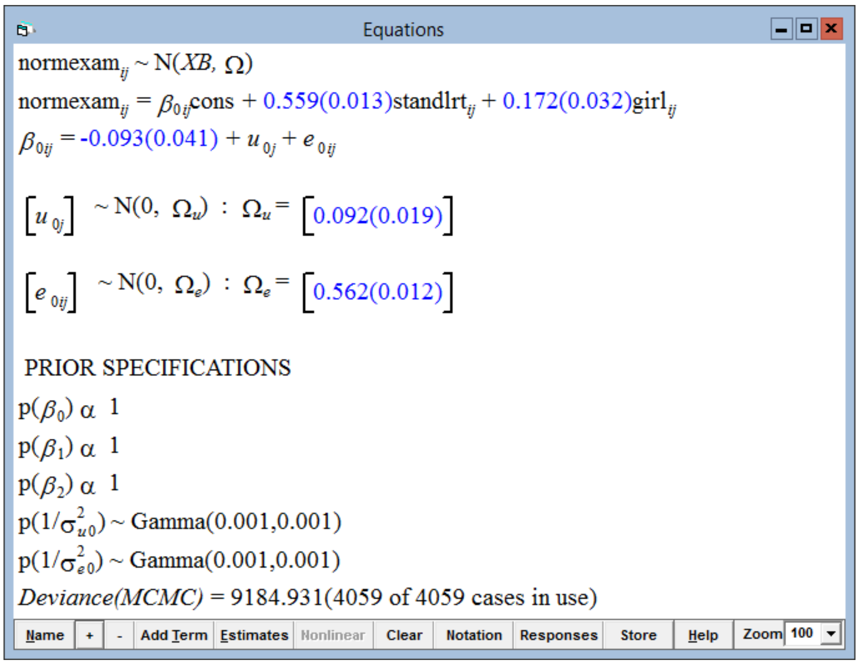
3.3 Comparison between fixed and random school effects

In the last chapter we fitted a fixed effects model that also included a gender effect. We can now add this gender term into our above model. To do this we need to do the following:

- Change estimation mode to IGLS.
- Click on the **Add term** button in the **Equations** window.
- Select **girl** from the **variable** list and click on the **Done** button.

- Click on the **Start** button to set up the model.
- Change estimation mode to MCMC.
- Click on the **Start** button to run the model using MCMC.

When the estimation has finished this time you should have estimates in the Equations window as follows:



$\text{normexam}_{ij} \sim \text{N}(XB, \Omega)$

$\text{normexam}_{ij} = \beta_{0ij}\text{cons} + 0.559(0.013)\text{standlrt}_{ij} + 0.172(0.032)\text{girl}_{ij}$

$\beta_{0ij} = -0.093(0.041) + u_{0j} + e_{0ij}$

$\begin{bmatrix} u_{0j} \end{bmatrix} \sim \text{N}(0,\ \Omega_u) \ : \ \Omega_u = \begin{bmatrix} 0.092(0.019) \end{bmatrix}$

$\begin{bmatrix} e_{0ij} \end{bmatrix} \sim \text{N}(0,\ \Omega_e) \ : \ \Omega_e = \begin{bmatrix} 0.562(0.012) \end{bmatrix}$

PRIOR SPECIFICATIONS

$p(\beta_0) \propto 1$

$p(\beta_1) \propto 1$

$p(\beta_2) \propto 1$

$p(1/\sigma^2_{u0}) \sim \text{Gamma}(0.001,0.001)$

$p(1/\sigma^2_{e0}) \sim \text{Gamma}(0.001,0.001)$

Deviance(MCMC) = 9184.931(4059 of 4059 cases in use)

Name + - Add Term Estimates Nonlinear Clear Notation Responses Store Help Zoom 100

If we were now to compare the DIC diagnostic for this model with the equivalent fixed effect formulation fitted at the end of the last chapter we would get the following:

Dbar	D(thetabar)	pD	DIC	
9184.93	9124.37	60.56	9245.49	*(random effects)*
9183.46	9115.36	68.10	9251.56	*(fixed effects)*

So we see here that in terms of fit (D(thetabar) column) the random effect model is actually a worse fit to the data than the fixed effects model. However due to the dependency between the random effects the actual effective number of parameters has been reduced by ≈ 7.5 so we have a less complex model. When this is taken into account the actual diagnostic DIC value for the random effects model suggests an improvement of 6 points.

In the next chapter we will continue looking at the random effects variance components model introduced in this chapter and consider other estimation methods and other features of the model

Chapter learning outcomes

- ⋆ What is meant by fixed effects, random effects and levels.
- ⋆ How to set up a variance components model
- ⋆ How to get more MCMC diagnostic information in MLwiN
- ⋆ How to compare the fit of random and fixed effect models.

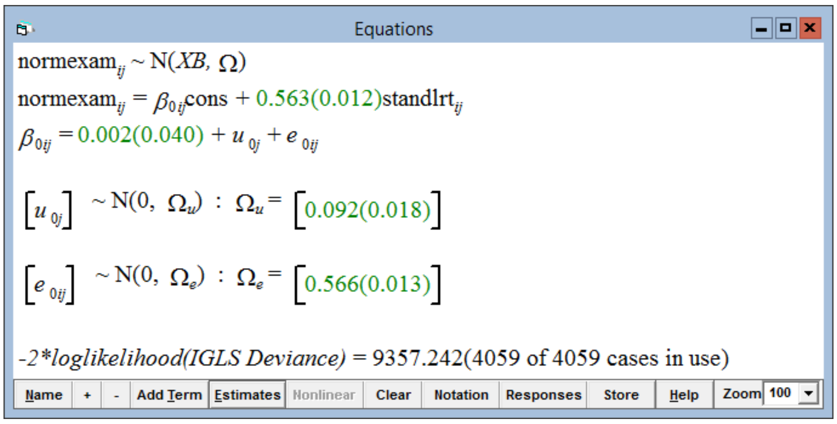
Chapter 4

Other Features of Variance Components Models

In this chapter we will return to the variance components model we considered in the last chapter. We will firstly show how to fit this model using the other MCMC methods available in MLwiN. We will then look at some other features of the variance components model for example residuals, school ranks and the intra-school correlation. The final model we considered in Chapter 3 had gender effects that we will now remove. To remove a term from a model we do the following:

- Select **IGLS** from the **Estimation** menu.
- Click on the **girl** variable in the **Equations** window.
- From the X variable screen that appears click on **Delete Term**.
- Run this model with the IGLS method by clicking on the **Start** button.

The **Equations** window should now look as follows:



$\text{normexam}_{ij} \sim \text{N}(XB, \Omega)$

$\text{normexam}_{ij} = \beta_{0ij}\text{cons} + 0.563(0.012)\text{standlrt}_{ij}$

$\beta_{0ij} = 0.002(0.040) + u_{0j} + e_{0ij}$

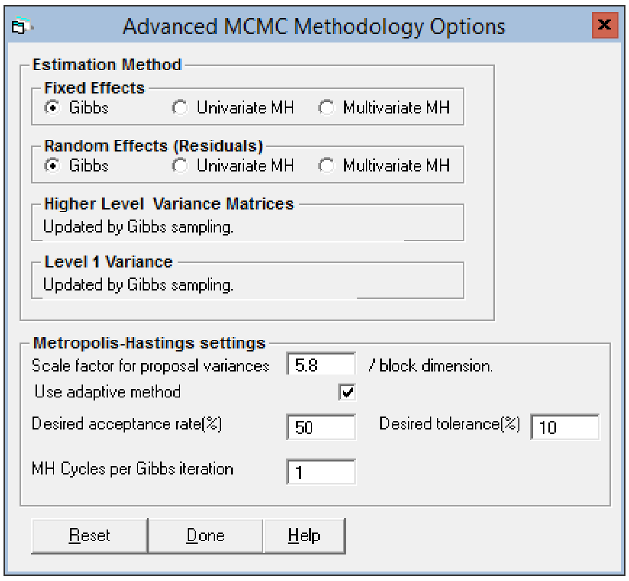
$\left[u_{0j}\right] \sim \text{N}(0,\ \Omega_u) \ : \ \Omega_u = \left[0.092(0.018)\right]$

$\left[e_{0ij}\right] \sim \text{N}(0,\ \Omega_e) \ : \ \Omega_e = \left[0.566(0.013)\right]$

*-2*loglikelihood(IGLS Deviance)* = 9357.242(4059 of 4059 cases in use)

4.1 Metropolis Hastings (MH) sampling for the variance components model

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 Methodology Options** window. Metropolis Hastings sampling is particularly useful for multilevel generalised linear models as will be seen in the later chapters of this manual. We shall firstly see how it can be used on a normal response variance components model. To use MH sampling, go back to the **Estimation control** window and click on **MCMC**. Then select **MCMC/MCMC methods** from the **Model** menu, which will bring up the following window:



Advanced MCMC Methodology Options

Estimation Method

Fixed Effects: [✓ Gibbs] Univariate MH Multivariate MH

Random Effects (Residuals): [✓ Gibbs] Univariate MH Multivariate MH

Higher Level Variance Matrices: Updated by Gibbs sampling.

Level 1 Variance: Updated by Gibbs sampling.

Metropolis-Hastings settings

Scale factor for proposal variances: 5.8 / block dimension.

Use adaptive method: ✓

Desired acceptance rate(%): 50 Desired tolerance(%): 10

MH Cycles per Gibbs iteration: 1

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 the fixed effects.
- Select the method **Univariate MH** for the random effects (residuals).
- Select the **Done** button.

We now look at how to use the settings.

4.2 Metropolis-Hastings settings

The performance of the Metropolis Hastings method depends very much on the proposal distribution used. A proposal distribution that accepts too many or too few proposals will produce highly autocorrelated chains. In MLwiN, the Metropolis Hastings sampler has some additional settings to help 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 & 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.

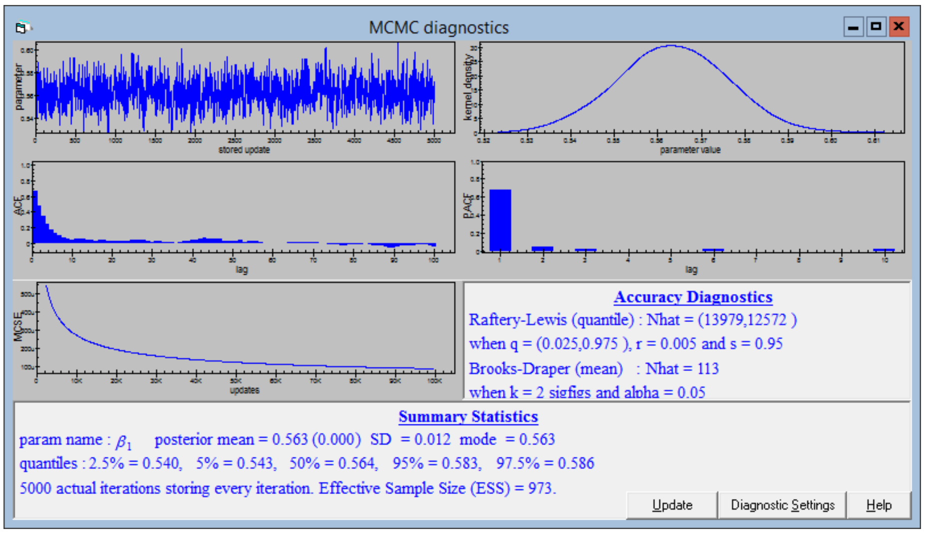
The second approach, the adaptive method, which is used by default in the development release of MLwiN is to find proposal distributions that give a particular desired acceptance rate. Experience suggests that rates of between 30% and 70% provide a useful compromise between a proposal variance that is 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, whereas 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 on the **Start** button, 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 modified every hundred iterations depending on the acceptance rate in the current batch of hundred iterations 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 variance components model with the adaptive method.

4.3 Running the variance components 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. 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:



Accuracy Diagnostics
Raftery-Lewis (quantile) : Nhat = (13979,12572)
when q = (0.025,0.975), r = 0.005 and s = 0.95
Brooks-Draper (mean) : Nhat = 113
when k = 2 sigfigs and alpha = 0.05

Summary Statistics
param name : β_1 posterior mean = 0.563 (0.000) SD = 0.012 mode = 0.563
quantiles : 2.5% = 0.540, 5% = 0.543, 50% = 0.564, 95% = 0.583, 97.5% = 0.586
5000 actual iterations storing every iteration. Effective Sample Size (ESS) = 973.

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 sampling 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 results in an effective sample size of only 973. This 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 16,400 for the Raftery-Lewis diagnostic for the 2.5% quantile.

The following table compares parameter estimates from IGLS, Gibbs and MH with both the scale factor and adaptive methods.

Parameter	IGLS	Gibbs	MH Scale Factor = 5.8	MH Adaptive with defaults
β_0	0.002	0.005	0.013	-0.005
β_1	0.563	0.563	0.563	0.563
σ^2_{u0}	0.092	0.097	0.097	0.097
σ^2_{e0}	0.566	0.566	0.566	0.566

MH and Gibbs sampling show good agreement, and apart from the level two 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 and in all cases this parameter is effectively zero. It is often useful with MCMC estimation to try both Gibbs and MH to confirm your estimates. We will now describe some additional MH features available in MLwiN.

4.4 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 reduces to 5,973 and the autocorrelation of its chain is reduced to 0.4 (ESS increased to 2,406). Note that this reduction in autocorrelation has to be balanced by the increase in time to run the model and in this case the 5000 iterations takes just under double the time.

4.5 Block updating MH sampling

As described earlier, 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 (for a variance components model there is only 1 residual per level 2 unit). In contrast the MH estimation method uses univariate updates for each parameter separately. In this development release, a block updating MH method is also available. This method updates the parameters in the same blocks used by the Gibbs sampling method.

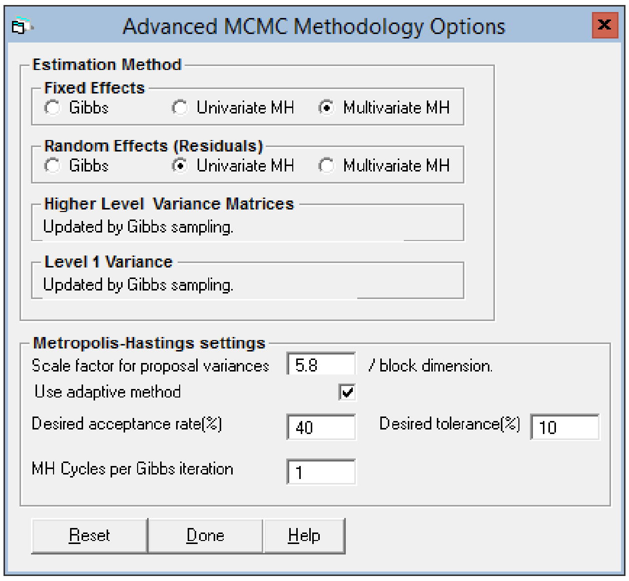
Parameters are updated in blocks using multivariate Normal proposals that take account of the correlation between the parameters in the block. As several parameters are updated together acceptance rates for each block will be lower than the acceptance rates achieved by updating each parameter individually, although updating the block should be faster than updating each parameter individually. The block updating sampler constructs a proposal based on the covariance matrix of the parameters in the block and a tuning constant. See Browne & Draper (2000) for more details.

In the variance components example, there is only one set of level 2 residuals but two 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 Methodology Options** window and change the settings as follows:

- Select the method **Multivariate MH** for the fixed effects.
- Change the **Desired acceptance rate** (%) to 40.
- Change the **MH cycles per Gibbs iteration** back to 1.

We have here changed the desired acceptance rate to 40% in the Metropolis Hastings settings box. This is because the optimal acceptance rate for a block update is smaller the larger the block size. The block updating method is currently regarded as 'experimental' because 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. Currently however the user may only enter a global desired acceptance rate.

The **Advanced MCMC Methodology Options** window should then look as follows:


Advanced MCMC Methodology Options

Estimation Method

Fixed Effects: Gibbs [✓ Multivariate MH] Univariate MH

Random Effects (Residuals): Gibbs [✓ Univariate MH] Multivariate MH

Higher Level Variance Matrices: Updated by Gibbs sampling.

Level 1 Variance: Updated by Gibbs sampling.

Metropolis-Hastings settings

Scale factor for proposal variances: 5.8 / block dimension.

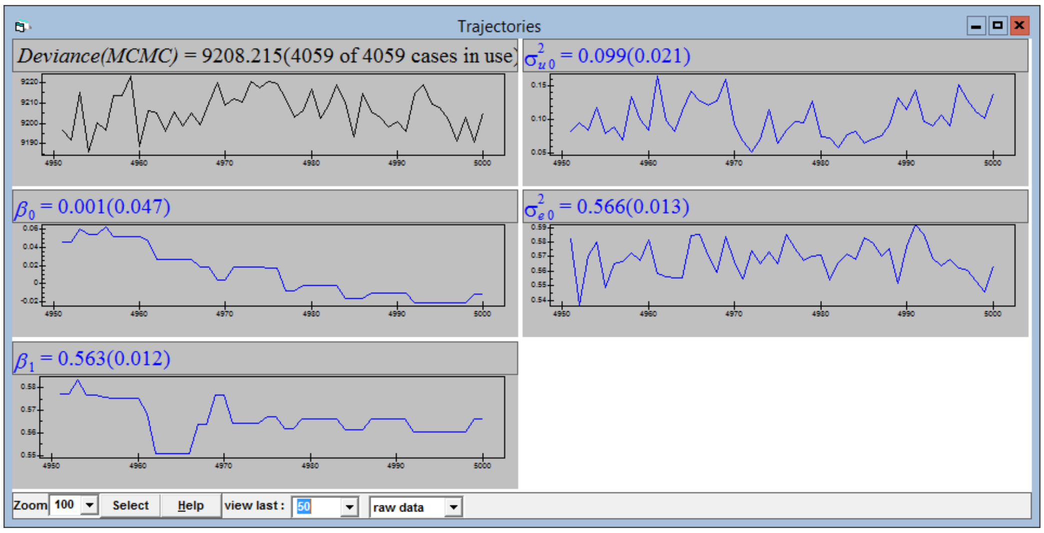
Use adaptive method: ✓

Desired acceptance rate(%): 40

Desired tolerance(%): 10

MH Cycles per Gibbs iteration: 1

The method was run for 5,000 iterations after an adapting period and trajectory traces of the last 50 iterations of the chains (which can be obtained by modifying the **view last** box) can be seen below:


Trajectories
Deviance(MCMC) = 9208.215(4059 of 4059 cases in use)
σ^2_{u0} = 0.099(0.021)
β_0 = 0.001(0.047)
σ^2_{e0} = 0.566(0.013)
β_1 = 0.563(0.012)
Select
Help

Zoom 100 view last: 50 raw data

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, as the jumps caused by accepted proposals occur simultaneously. Note that sometimes (for example β_0 in the last iteration) the accepted new value is very similar to the current value. We will now return to Gibbs sampling and consider other features of the variance components model. First we must reset all of the MCMC settings:

- Select **IGLS** and run the model (press the **Start** button).
- Change estimation mode to **MCMC**
- Select **MCMC/MCMC Methods** from the **Model** menu.
- Click on the **Reset** button.
- Click on the **Done** button.

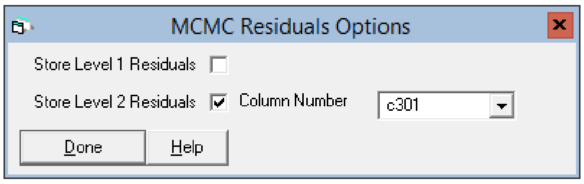
4.6 Residuals in MCMC

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, only means and standard errors are stored from their chains. It is then these values that are used when the residuals options, as demonstrated in the User's Guide to MLwiN Chapter 3, 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:

- Select **MCMC/Store Residuals** from the Model menu.
- The **MCMC Residuals Options** window will appear.
- Click in the box to **Store Level 2 Residuals**.

and the window will look as follows:

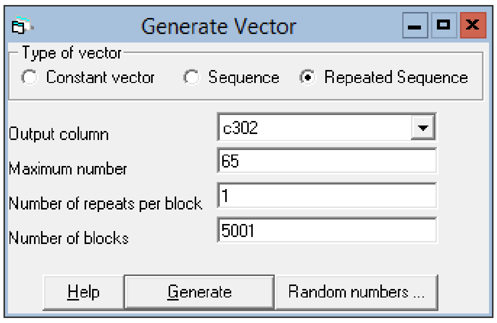


Store Level 1 Residuals [ ]
Store Level 2 Residuals [✓] Column Number c301

This option means that the chain values for the 65 residuals will be stacked in column **c301**.

- Click on the **Done** button.
- Run the model using Gibbs sampling as before except this time set the length of **monitoring chain** to 5001 to make calculation of quantiles easier.

After running the model we will now have to split column **c301** 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:



Type of vector
Constant vector Sequence [✓ Repeated Sequence]

Output column c302
Maximum number 65
Number of repeats per block 1
Number of blocks 5001

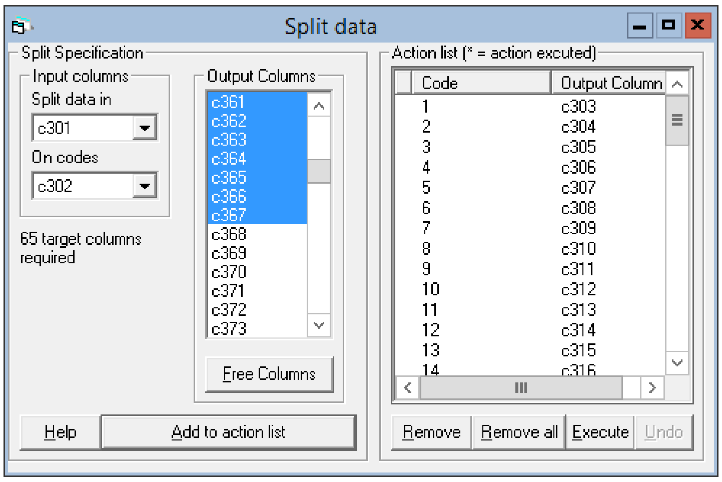
- Click on the **Generate** button to create the column.

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:

- Select **Split column** from the **Data Manipulation** menu.

- For input columns, select **c301** as the data column and
- Select **c302** as the code column.
- Select **c303-c367** as the output columns (use the Shift key or the Ctrl Key along with the left mouse button to select multiple columns).
- Click on the **Add to action list** button.

The above set of instructions will then produce the following screen:

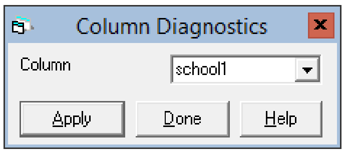


Code	Output Column
1	c303
2	c304
3	c305
4	c306
5	c307
6	c308
7	c309
8	c310
9	c311
10	c312
11	c313
12	c314
13	c315
14	c316

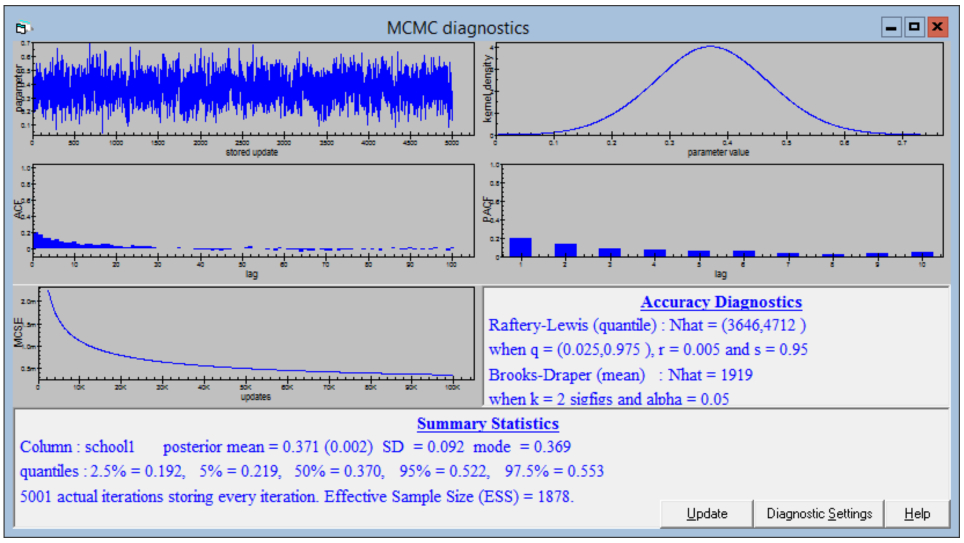
- Click on **Execute** to run the command

The columns **c303-c367** will now contain the chains of the school level residuals.

We can name the columns **c303-c367** if we wish 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 as follows:



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:



Accuracy Diagnostics
Raftery-Lewis (quantile) : Nhat = (3646,4712)
when q = (0.025,0.975), r = 0.005 and s = 0.95
Brooks-Draper (mean) : Nhat = 1919
when k = 2 sigfigs and alpha = 0.05

Summary Statistics
Column : school1 posterior mean = 0.371 (0.002) SD = 0.092 mode = 0.369
quantiles : 2.5% = 0.192, 5% = 0.219, 50% = 0.370, 95% = 0.522, 97.5% = 0.553
5001 actual iterations storing every iteration. Effective Sample Size (ESS) = 1878.

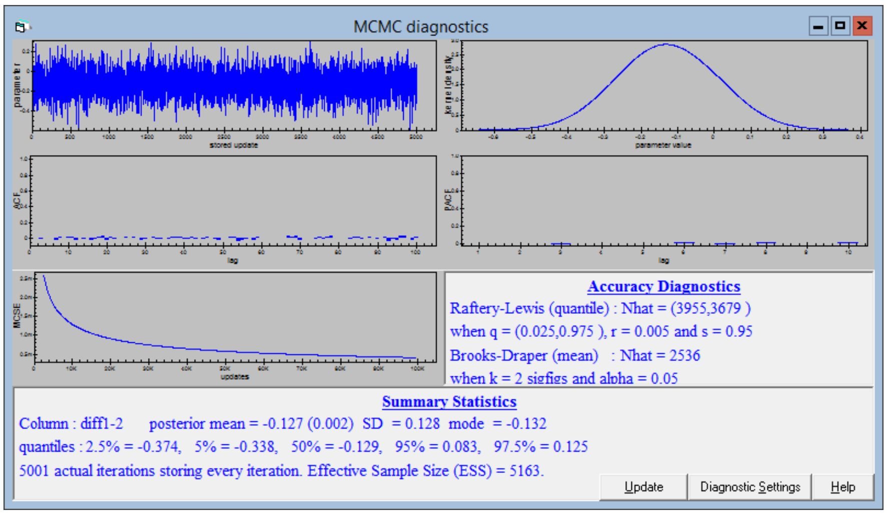
As can be seen from the kernel density plot of the residual, this residual has a posterior distribution that is close to Normal, which is what we would expect. This means that we can generate an interval estimate based on the Normal assumption and do not need to use the quantiles.

4.7 Comparing two schools

We may also be interested in two schools (for example schools 1 and 2) and finding which school has the larger effect in our model. We could do this by simply comparing the residuals and the standard errors for these two schools. Alternatively we could look at the chain of differences, which can be calculated by typing the following commands in the **Command interface** window:

```
▶ Calc c368=c303-c304
▶ Name c368 'Diff1-2'
```

Then we can look at a plot of this new function using the column diagnostics window, which will give the following diagnostics:



Accuracy Diagnostics
Raftery-Lewis (quantile) : Nhat = (3955,3679)
when q = (0.025,0.975), r = 0.005 and s = 0.95
Brooks-Draper (mean) : Nhat = 2536
when k = 2 sigfigs and alpha = 0.05

Summary Statistics
Column : diff1-2 posterior mean = -0.127 (0.002) SD = 0.128 mode = -0.132
quantiles : 2.5% = -0.374, 5% = -0.338, 50% = -0.129, 95% = 0.083, 97.5% = 0.125
5001 actual iterations storing every iteration. Effective Sample Size (ESS) = 5163.

Here we can see that the value 0 is inside both the 90% and 95% intervals and so the two school residuals are not significantly different, although on average school 2 is performing better than school 1.

We can again use the command interface window to see how often each school performs better in these chains with the following commands:

```
▶ Calc c369 = 'diff1-2' > 0

▶ Aver c369
```

These two commands give the average 0.166 so that school 1 performs better than school 2 in 16.6% of the iterations. This can be compared with the 1-sided P-value of 0.161 when testing whether this parameter is significantly different from zero (assuming Normality).

4.8 Calculating ranks of schools

We may also be interested in the ranks of the schools rather than their actual residuals and MCMC allows us to calculate point and interval estimates for each school's rank (see Goldstein & Speigelhalter, 1996).

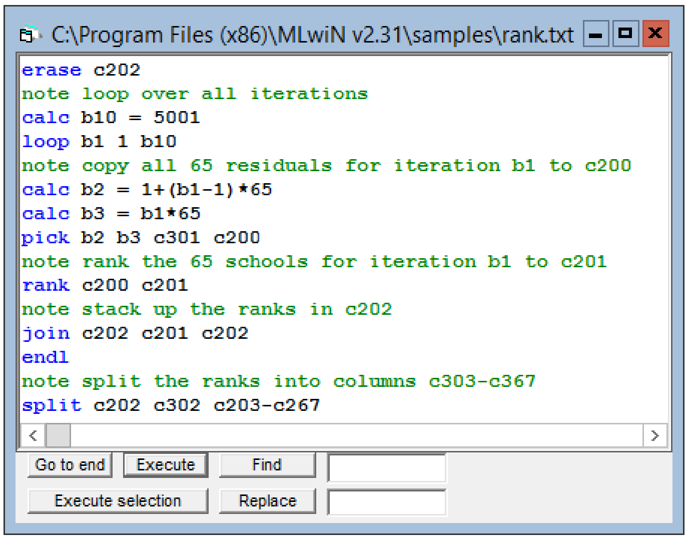
Here we need to run several commands to split and sort the residuals and then rank the schools and so we have included the required commands in the macro **rank.txt**.

To open this macro:

- Select **Open Macro** from the **File** menu.

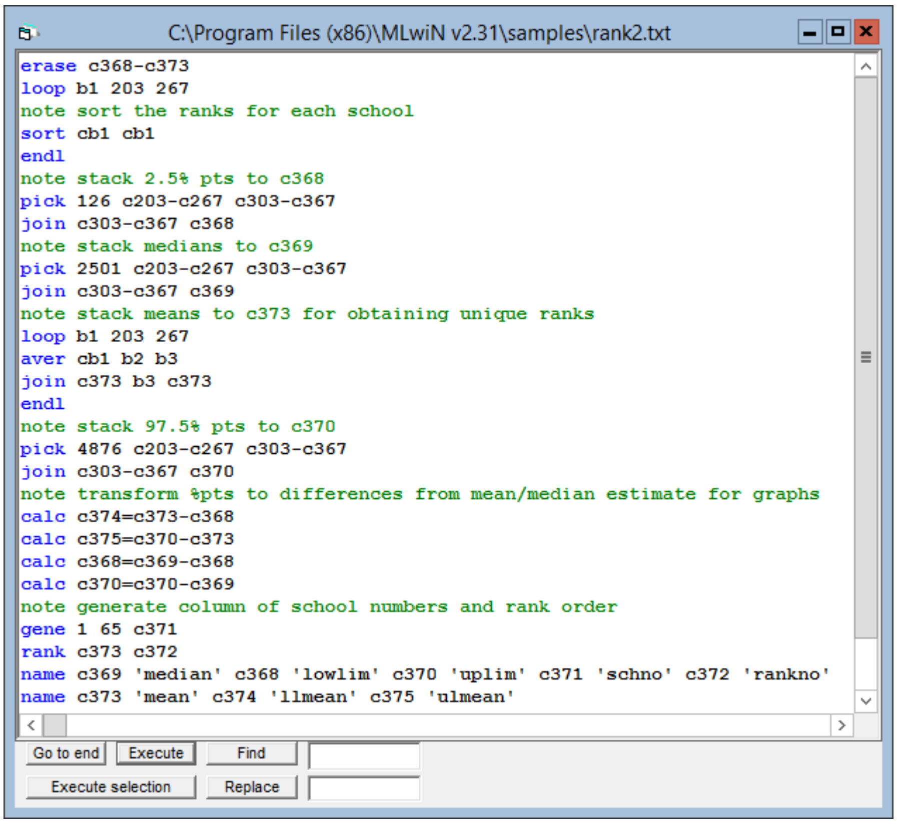
- Select **rank.txt** from the list of files.
- Click on the **Open** button.

The macro will then appear as follows:



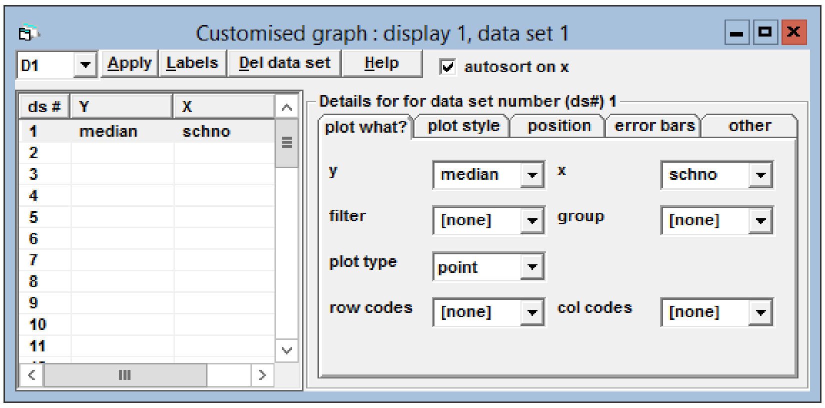
```
erase c202
note loop over all iterations
calc b10 = 5001
loop b1 1 b10
note copy all 65 residuals for iteration b1 to c200
calc b2 = 1+(b1-1)*65
calc b3 = b1*65
pick b2 b3 c301 c200
note rank the 65 schools for iteration b1 to c201
rank c200 c201
note stack up the ranks in c202
join c202 c201 c202
endl
note split the ranks into columns c303-c367
split c202 c302 c203-c267
```

This macro will give chains for the ranks of each school in columns **c203-c267**. Run this macro now by clicking on the **Execute** button. Note that this macro will take a bit of time to run. It would be useful from this to calculate a ‘caterpillar’ style plot for the ranks and the following macro **rank2.txt** (which can be opened and run after the **rank.txt** macro) calculates the median, 2.5% and 97.5% quantiles for each school.



```
erase c368-c373
loop b1 203 267
note sort the ranks for each school
sort cb1 cb1
endl
note stack 2.5% pts to c368
pick 126 c203-c267 c303-c367
join c303-c367 c368
note stack medians to c369
pick 2501 c203-c267 c303-c367
join c303-c367 c369
note stack means to c373 for obtaining unique ranks
loop b1 203 267
aver cb1 b2 b3
join c373 b3 c373
endl
note stack 97.5% pts to c370
pick 4876 c203-c267 c303-c367
join c303-c367 c370
note transform %pts to differences from mean/median estimate for graphs
calc c374=c373-c368
calc c375=c370-c373
calc c368=c369-c368
calc c370=c370-c369
note generate column of school numbers and rank order
gene 1 65 c371
rank c373 c372
name c369 'median' c368 'lowlim' c370 'uplim' c371 'schno' c372 'rankno'
name c373 'mean' c374 'llmean' c375 'ulmean'
```

Note that this macro will reuse the columns in which the 65 residual chains were stored earlier. We can then graph the schools by using the **Customised graph** window; we first set up the x and y variables as follows:



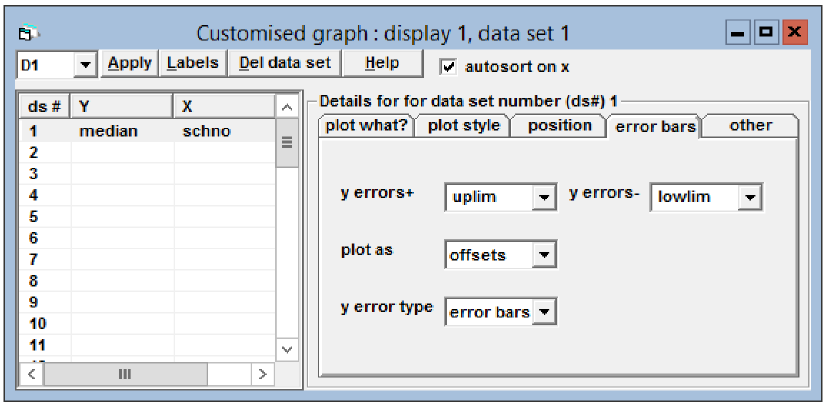
autosort on x [✓]

ds #	Y	X
1	median	schno
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		

Details for for data set number (ds#) 1

y: median
x: schno
filter: [none]
group: [none]
plot type: point
row codes: [none]
col codes: [none]

Next we set up the limits on the **error bars** tab as follows:



Customised graph : display 1, data set 1

D1 | Apply | Labels | Del data set | Help | ☑ autosort on x

ds #	Y	X
1	median	schno
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		

Details for for data set number (ds#) 1

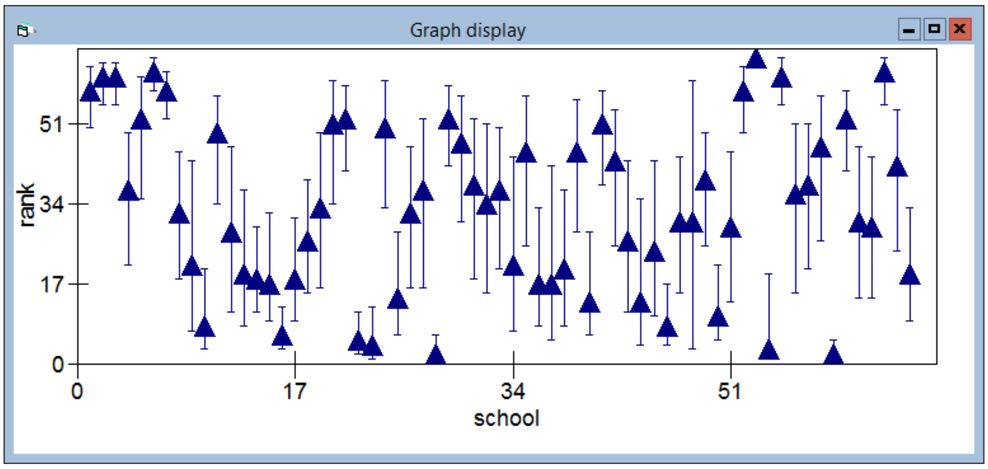
plot what? | plot style | position | error bars | other

y errors+ uplim y errors- lowlim

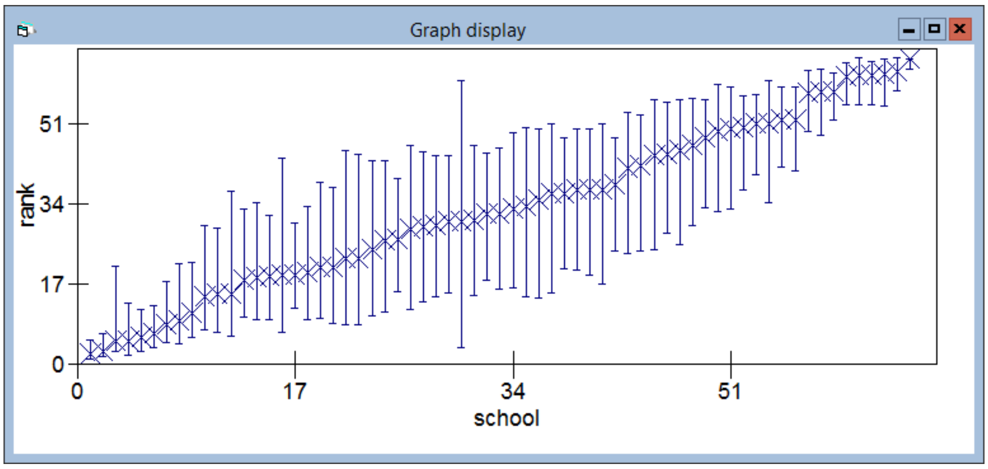
plot as offsets

y error type error bars

Then the graph will look as follows (with titles added) once the **Apply** button has been pressed:



This graph shows the ranks (plus intervals) for all 65 schools, noting that rank 65 here is the highest achieving school and rank 1 the lowest (after adjusting for intake score). We can see that there is often great overlap between the relative rankings of the schools and in many cases great uncertainty in the actual ranks. We can convert the graph into the more common 'caterpillar' plot by replacing the '*y*' variable with '*mean*', the '*x*' variable with '*rankno*'. This sorts the schools according to rank rather than numeric order. The error bars columns should be replaced with the columns '*ulmean*' and '*llmean*'. Note that the mean has been used to calculate ranks (to avoid ties) so the points plotted are now mean ranks and not median ranks as in the previous graph. Note also that we have rescaled the axes and changed the symbol used for the points in this graph.



4.9 Estimating a function of parameters

We have now seen how to calculate accurate interval estimates for residuals and for ranks of the schools. There are however other parameters not estimated directly as part of the model that could be of interest. In Chapter 2 of the MLwiN User's Guide, for example, the intra-school correlation, ρ_s, was described. This parameter is a function of the level 1 and level 2 variance parameters and so can be calculated from these parameters via the simple formula:

$$\rho_s = \sigma_u^2/(\sigma_u^2 + \sigma_e^2)$$

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. You should at this point have a Gibbs sampler run of 5001 iterations. If not, run the Gibbs sampler again using the default settings except for running for 5,001 iterations. All the parameters that can be viewed in the **Trajectories** window are stored in a stacked column (in this case **C1090** is always used) in a similar way to how the residuals were stored in the last section.

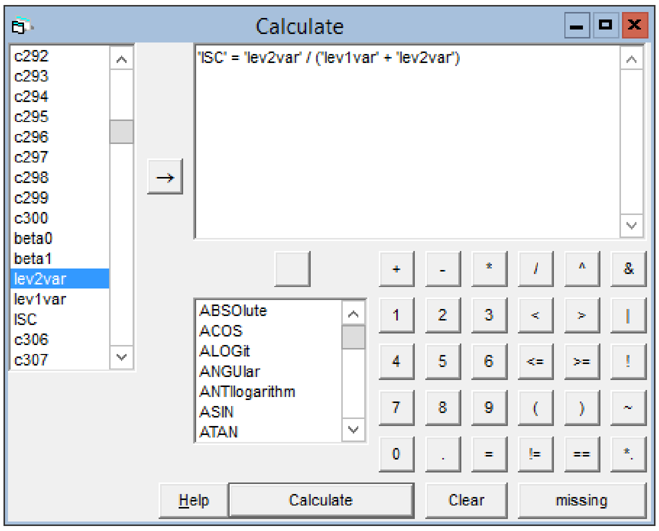
In order to calculate the function of parameters we are interested in we will have to firstly unstack column **c1090**. This can be done using the **Generate vector** and **Split column** windows from the **Data manipulation** menu in a similar way to the residuals example in the previous section. Alternatively the **Command interface** window can be used and the following commands entered:

```
▶ code 4 1 5001 c300
▶ split c1090 c300 c301-c304
```

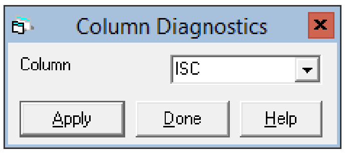
Having split the variables into different columns we can then name the columns either by using the **Names** window or again by using the **Command Interface** window by typing the NAME command as follows:

```
▶ name c301 'beta0' c302 'beta1' c303 'lev2var' c304
  'lev1var' c305 'ISC'
```

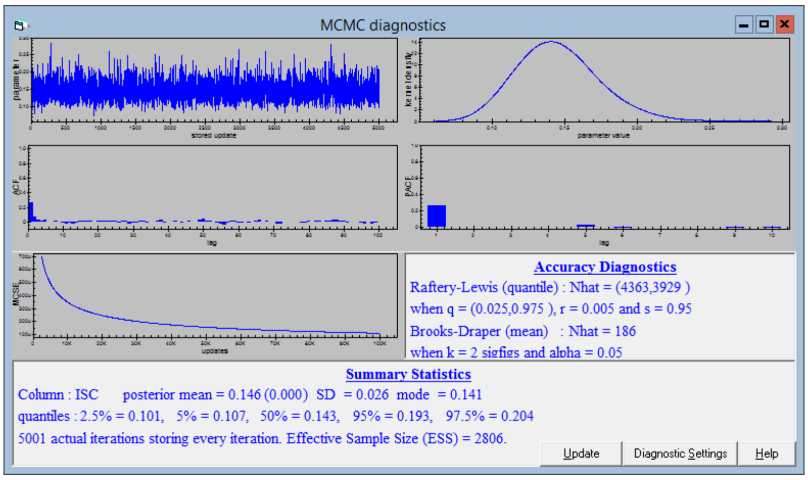
We now need to calculate the chain of values for the intra-school correlation (ISC) and we do this by using the **Calculate** window, which can be found in the **Data Manipulation** menu. The column 'ISC' should be calculated as follows:


'ISC' = 'lev2var' / ('lev1var' + 'lev2var')

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:


Column ISC

Having clicked on **Apply** the diagnostics window should appear as follows:



Accuracy Diagnostics
Raftery-Lewis (quantile) : Nhat = (4363,3929)
when q = (0.025,0.975), r = 0.005 and s = 0.95
Brooks-Draper (mean)   : Nhat = 186
when k = 2 sigfigs and alpha = 0.05

Summary Statistics
Column : ISC   posterior mean = 0.146 (0.000) SD = 0.026 mode = 0.141
quantiles : 2.5% = 0.101, 5% = 0.107, 50% = 0.143, 95% = 0.193, 97.5% = 0.204
5001 actual iterations storing every iteration. Effective Sample Size (ESS) = 2806.

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.

Chapter learning outcomes

- ⋆ How to change MCMC estimation method from Gibbs sampling to MH sampling for some steps of the algorithm.
- ⋆ What other MCMC settings there are and what they do.
- ⋆ How to store residual chains
- ⋆ How to calculate ranks of schools
- ⋆ How to calculate estimates and chains for derived variables

Chapter 5

Prior Distributions, Starting Values and Random Number Seeds

In this chapter we consider some other features of the MCMC estimation procedures in MLwiN. We will still consider the variance components model with one predictor (**standlrt**) discussed in the last two chapters and will look at how to modify the prior distributions, starting values and random number seeds used for this model.

5.1 Prior distributions

In Chapter 1 we described the default prior distributions used in MLwiN. We also mentioned that these defaults are different (for the variance parameters) from the priors used in the first version of MLwiN (release 1.0). This is because these new priors generally give less positive bias when the parameter estimate based on the mean is used. The old default priors can still be selected via the **MCMC** window available from the **Model** menu, as can informative priors.

5.2 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 $p(\Omega) \propto 1$ (a constant prior over the positive definite matrices Ω, or a uniform prior for σ^2 for a single variance)

These priors are functionally equivalent to the following proper priors:

- For single 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.

- To use these priors select **Uniform on variance scale** for the default diffuse priors for variance parameters on the **MCMC priors** window available from the **Model** menu.

Comparing these priors (with the default $\Gamma^{-1}(\varepsilon, \varepsilon)$ priors used thus far for single variances) using the Gibbs sampler on the variance components model we get the following results (using a monitoring run of 5,000):

Parameter	IGLS	Gibbs	
		($\Gamma^{-1}(\varepsilon, \varepsilon)$ priors)	(Uniform priors)
β_0	0.002 (0.040)	0.005 (0.042)	0.004 (0.042)
β_1	0.563 (0.012)	0.563 (0.012)	0.563 (0.013)
σ^2_{u0} (Mean)	0.092 (0.018)	0.097 (0.021)	0.101 (0.022)
σ^2_{u0} (Mode)	-	0.092	0.095
σ^2_{e0}	0.566 (0.013)	0.566 (0.013)	0.566 (0.013)

So we see that the Uniform prior tends to give larger variance estimates than the default priors when the number of level 2 units is small. Browne (1998) and Browne & Draper (2006) show this in more detail via simulation experiments and we will discuss running simulations in MLwiN in greater detail in Chapter 8.

In this version of MLwiN it is also possible to change the parameters of the Γ^{-1} priors via the **MCMC/ priors** window available from the **Model** menu. The defaults are $a = 0.001$, $b = 0.001$ but another popular choice is to set $a = 1.0$ and $b = 0.001$.

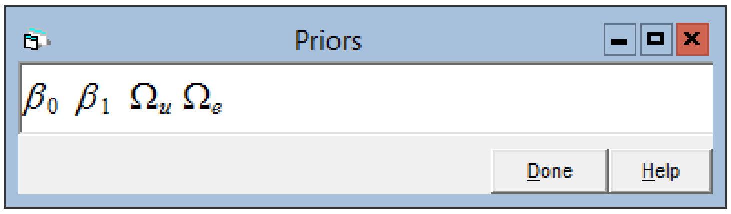
5.3 Using informative priors

MLwiN also allows the user to specify informative priors for any of the parameters in the model. This could be useful if the user already has some prior knowledge on the values of the unknown parameters. We will firstly consider specifying an informative prior for the fixed effect associated with the intake score (LRT).

Again we will start with IGLS starting values so run the model using the IGLS method. Then on the MCMC Priors window:

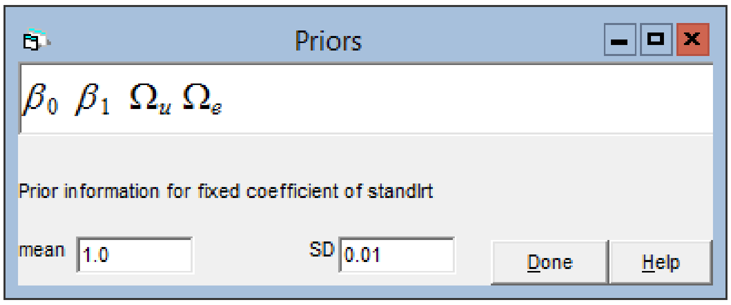
- Click on **Gamma priors** to return to the default variance priors.
- Click on the **Informative Priors...** button.

The following window will appear showing all the parameters in the model (in this case the level 1 and level 2 covariance matrices contain just a single variance term):



For the fixed parameters, informative 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. Note that for scalar variances, as in this example, an informative inverse Gamma distribution will be used. We wish to add a prior for the slope parameter, β_1. Let us assume, for illustration, that from a previous study we have (after transformations) observed an estimated coefficient of 1.0 for LRT intake score.

If you click on, for example, β_1, and then enter an informative prior with a mean of 1 (remembering the posterior estimate from a 'diffuse' prior 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:



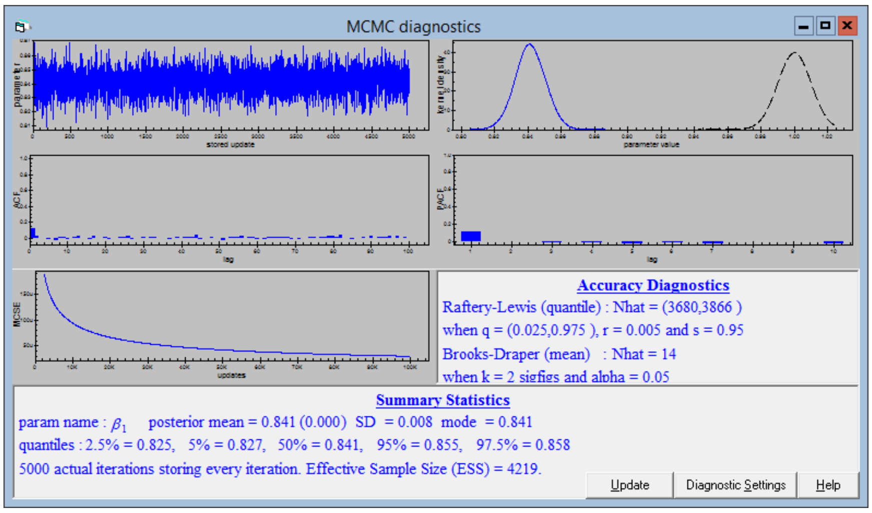
Prior information for fixed coefficient of standlrt

mean 1.0 SD 0.01

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 the **Done** buttons to close down the two **Priors** windows.
- Click on the **Start** button 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:

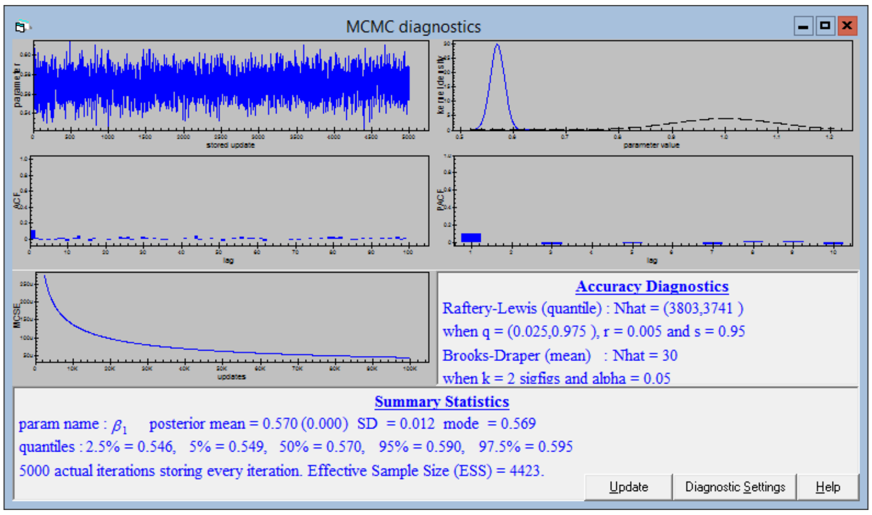


Accuracy Diagnostics
Raftery-Lewis (quantile) : Nhat = (3680,3866)
when q = (0.025,0.975), r = 0.005 and s = 0.95
Brooks-Draper (mean) : Nhat = 14
when k = 2 sigfigs and alpha = 0.05

Summary Statistics
param name : β_1 posterior mean = 0.841 (0.000) SD = 0.008 mode = 0.841
quantiles : 2.5% = 0.825, 5% = 0.827, 50% = 0.841, 95% = 0.855, 97.5% = 0.858
5000 actual iterations storing every iteration. Effective Sample Size (ESS) = 4219.

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 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 0.1 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 as is shown in the kernel plot in the diagnostics below:



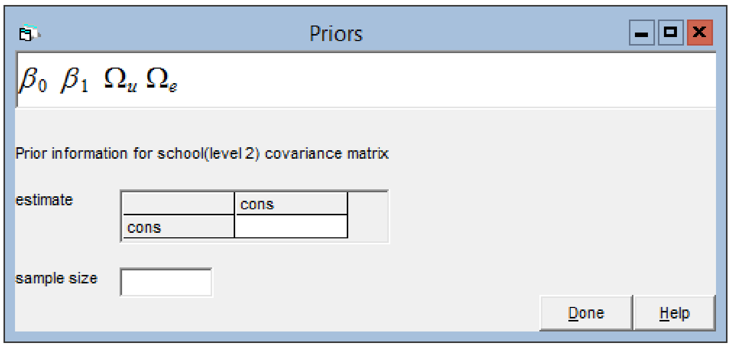
Accuracy Diagnostics
Raftery-Lewis (quantile) : Nhat = (3803,3741)
when q = (0.025,0.975), r = 0.005 and s = 0.95
Brooks-Draper (mean) : Nhat = 30
when k = 2 sigfigs and alpha = 0.05

Summary Statistics
param name : β_1 posterior mean = 0.570 (0.000) SD = 0.012 mode = 0.569
quantiles : 2.5% = 0.546, 5% = 0.549, 50% = 0.570, 95% = 0.590, 97.5% = 0.595
5000 actual iterations storing every iteration. Effective Sample Size (ESS) = 4423.

In this example there is a distinct difference in the prior value for β_1 and the estimate from this dataset. If however the prior and data are more in concordance then including an informative prior will reinforce the data estimate by reducing the standard error of the estimate. Note that changing the structure of the current model will result in all informative prior information being erased.

5.4 Specifying an informative prior for a random parameter

The procedure for specifying informative priors for random parameters is somewhat different. Clear the existing prior on the slope coefficient (by clicking on β_1 and setting the two prior parameters to zero) and then click on the level two variance matrix Ω_u. The Priors window will then look as follows:



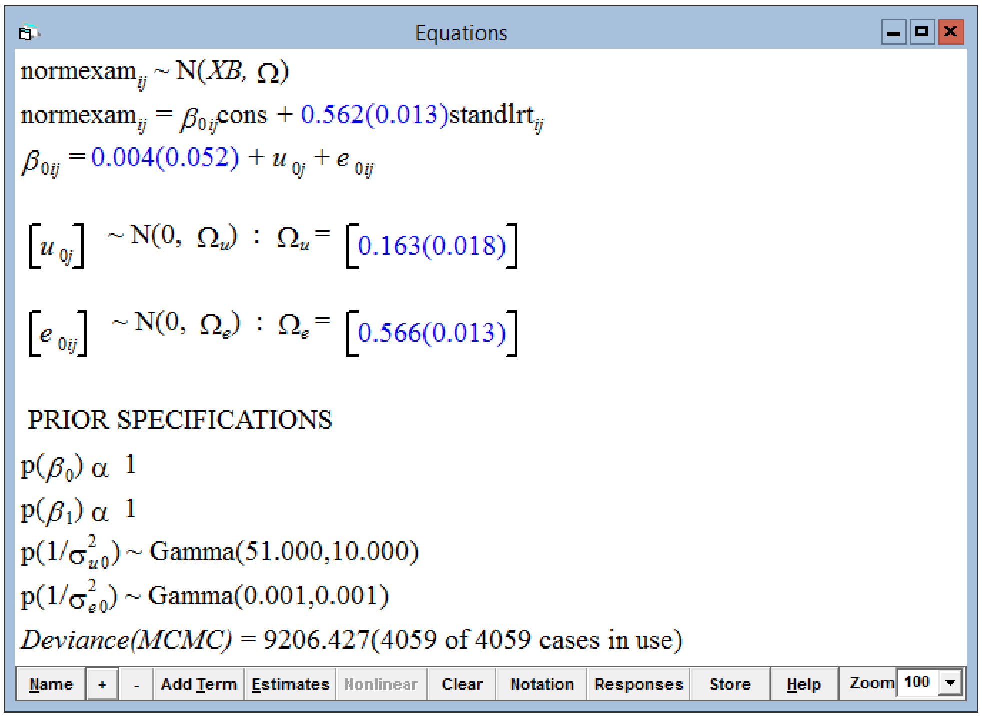
Prior information for school(level 2) covariance matrix

estimate

	cons
cons	

sample size

For illustration let us assume we have a prior estimate of the level 2 variance equal to 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. MLwiN will now convert this information into an Inverse Gamma prior for the variance as illustrated in the following **Equations** window obtained after running the model for 5,000 iterations. Note that you may need to press the + button to get prior information in the window.



$\text{normexam}_{ij} \sim \text{N}(XB, \Omega)$

$\text{normexam}_{ij} = \beta_{0ij}\text{cons} + 0.562(0.013)\text{standlrt}_{ij}$

$\beta_{0ij} = 0.004(0.052) + u_{0j} + e_{0ij}$

$\left[u_{0j}\right] \sim \text{N}(0, \Omega_u) : \Omega_u = \left[0.163(0.018)\right]$

$\left[e_{0ij}\right] \sim \text{N}(0, \Omega_e) : \Omega_e = \left[0.566(0.013)\right]$

PRIOR SPECIFICATIONS

$p(\beta_0) \alpha\ 1$

$p(\beta_1) \alpha\ 1$

$p(1/\sigma^2_{u0}) \sim \text{Gamma}(51.000,10.000)$

$p(1/\sigma^2_{e0}) \sim \text{Gamma}(0.001,0.001)$

Deviance(MCMC) = 9206.427(4059 of 4059 cases in use)

Here we see that the estimated value of the level two variance after 5,000 iterations is now 0.163 — fairly close to a weighted average of the estimate obtained with a diffuse prior and the informative prior estimate (weighted by the number of level two units the estimates are based on) — and the other parameter estimates are hardly changed. Currently in MLwiN the prior density is not shown for random parameters in the kernel plots. Before going on to the next session we should remove the informative priors and this is done by bringing up the **Informative priors** window and typing 0 for the **sample size** parameter.

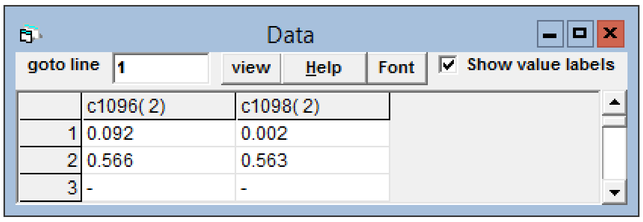
5.5 Changing the random number seed and the parameter starting values

The prior distributions described above actually change the model structure when fitted using MCMC sampling. The parameters we describe in this section do NOT change the form of the model but due to the stochastic nature of the MCMC methods slightly different estimates may result from modifying them. 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 reach similar 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 specific columns in the MLwiN worksheet. The starting values for the residuals are then produced by MLwiN calculating the maximum likelihood estimates for these parameters, conditional on the values of the fixed effects and variance parameters (stored in columns **c1098** and **c1096** respectively).

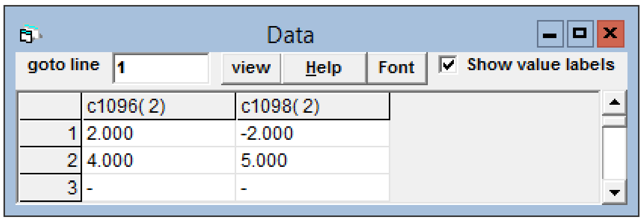
We will consider again our 2 level variance components model and run IGLS on the model. If you then open the **Data** window as described below you will see the following window.

- Select **View or Edit Data** from the **Data Manipulation** menu.
- Click on the **View** button.
- Select columns **c1096** and **c1098** (use the CTRL button to select both).
- Click on **OK**.



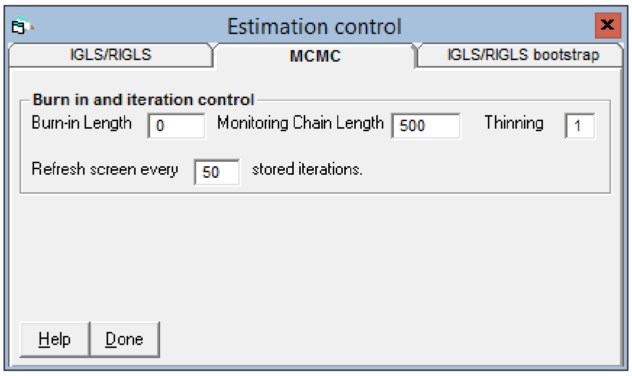
	c1096(2)	c1098(2)
1	0.092	0.002
2	0.566	0.563
3	-	-

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:



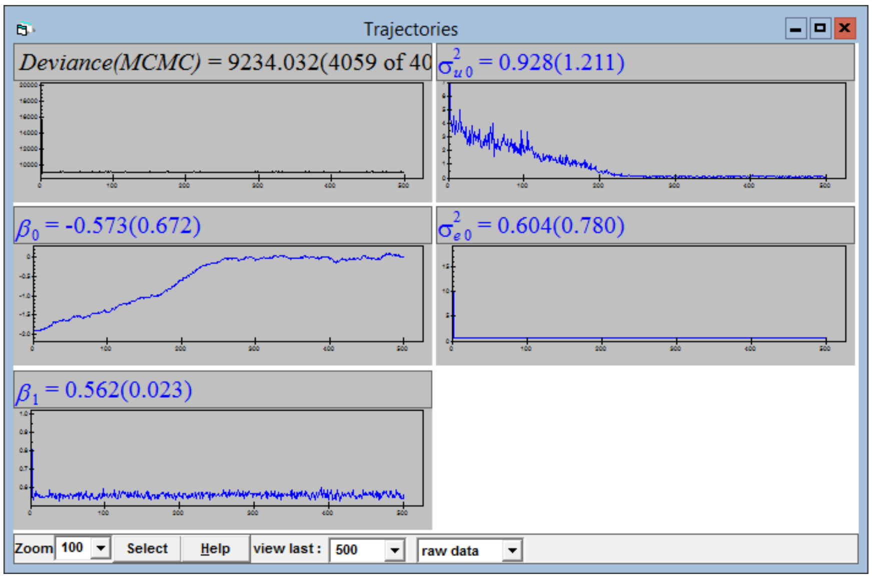
	c1096(2)	c1098(2)
1	2.000	-2.000
2	4.000	5.000
3	-	-

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. It is however possible by using the **Command interface** window to alter the values of the other parameters 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.



Burn in and iteration control
Burn-in Length 0 Monitoring Chain Length 500 Thinning 1
Refresh screen every 50 stored iterations.

If we then click on **Start**, the chains for the first 500 iterations can be seen in the following **Trajectories** window:



$Deviance(MCMC)$ = 9234.032(4059 of 40

σ^2_{u0} = 0.928(1.211)

β_0 = -0.573(0.672)

σ^2_{e0} = 0.604(0.780)

β_1 = 0.562(0.023)

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 (by changing the **monitoring chain length** to 1000 and clicking on the **More** button) 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 uni-modal posterior distributions. In some of the new models in this release of MLwiN this may not be guaranteed. If, however, it is already known that the posterior distributions should be uni-modal it is best to utilise the 'good' starting values obtained by IGLS, particularly when using MH sampling which may take longer to reach equilibrium (i.e. the point where it is actually sampling from the correct posterior distribution).

The random number seed can be set on the **MCMC/Random Number**

Seed window available from the **Model** menu. 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. Note that in MLwiN the random seed for the MCMC options is different from the random number seed used by the macro command language that can be set by the **SEED** command. The MCMC seed can be set by the **MCRS** command. To illustrate this behaviour the following table contains the point estimates (for sets of 5000 iterations after burnins of 500) 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^2_{u0}	0.097	0.097	0.097	0.097
σ^2_{e0}	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 making use of the options in this section is not generally required to ensure good MCMC performance. We include them only for completeness.

5.6 Improving the speed of MCMC Estimation

One feature of MCMC estimation methods is that they are computationally intensive, and generally take far longer to run than the likelihood-based 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: first to minimise the number of iterations required to give accurate estimates, and second 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.

We are continually trying to speed up the MCMC estimation procedures in

MLwiN and you should find that the speed of estimation is generally better than some other general purpose Bayesian modelling software. With the speed of computer chips also improving at an incredible rate the time taken by MCMC methods is continually improving.

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 666MHz PC, running the variance components model for 5,000 iterations after a burn in of 500.

Screen Format	Time
No windows	20 seconds
Equations window	25 seconds
Trajectories window	34 seconds
Both windows	38 seconds

As can be seen, displaying the windows — particularly the **Trajectories** window — slows the estimation down.

Chapter learning outcomes

- ⋆ How to change the default variance prior distributions.
- ⋆ How to specify informative prior distributions.
- ⋆ How to modify both the parameter starting values and random number seed.
- ⋆ How to speed up the methods.

Chapter 6

Random Slopes Regression Models

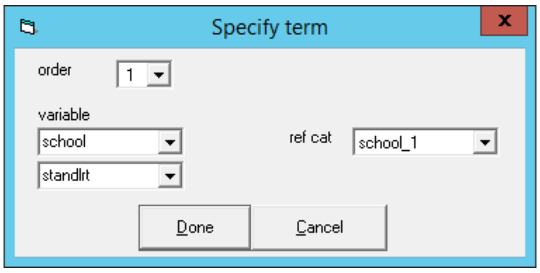
In the past three chapters we have considered the variance components model and looked at how we can apply different MCMC methods and priors to this model. We have also looked at some of the features of the model such as residuals and school ranks that can be calculated by the MCMC methods. We saw that compared to the single level models from Chapter 2 the variance components model—which has random intercepts—fits the data better. We will now continue our exploration of the dataset by considering fitting both random intercepts and slopes.

In Chapter 2 we saw that it is possible to account for school effects by fitting an intercept and a fixed term for each school (with the constraint that the fixed effect associated with school 1 is equal to 0) and that this results in a model with 66 fixed effects. It is also possible to fit a fixed effect model that accounts for both different school effects and different effects of intake score (LRT) for each school. This involves fitting the 'interaction' of school effect with LRT score and we will then have the two constraints that both the fixed effect associated with school 1 and the intake effect for school 1 are constrained to be zero. Note of course that this does not mean that school 1 has zero effects. School 1 is just the baseline school with regression line explained by the common intercept and slope terms, and the other school effects and LRT effects are then relative to this baseline school. To fit this model in MLwiN we need to again use the **Add Term** button on the **Equations** window. We first set up the model with individual school effects as described at the end of Chapter 2 but this time we will remove the effect of gender. Note that if you are following on from the last chapter you should also ensure the random number seed is set back to 1.

- In the **Equations** window click on β_2 (**girl**) and click on the **delete term** button.

- Click on the **Add Term** button.
- Select 1 in the **order** box.
- Select **school** as the first variable (the first box under variable).
- Select **standlrt** as the second variable.

The window should look as follows:



Now clicking on the **Done** button will add the 64 school by intake score interaction terms.

This will now have set up the model with 130 fixed effects. If you have the **Equations** window open you will notice that all the school × LRT terms have been added to the model.

- Click the **Start** button.

This will run the model using ordinary least squares and give estimates for the 130 fixed effects. Note that the school × LRT terms can be thought of as slope differences when comparing these 64 schools to the base school. If we were to re-parameterise the model by removing the global intercept and slope and instead add the intercept and slope indicators for the base school we will get exactly the same model but this time the parameters will be the actual intercepts and slopes for each school. We are therefore essentially, in this model, fitting separate regression lines to each school. To fit this model in MCMC we need to:

- Select the **MCMC** tab on the **Estimation Control** window.
- Click on the **Start** button.

This model has 130 fixed effects and so will take a few minutes to run. If we were to check the DIC diagnostic for this model we would obtain:

Dbar	D(thetabar)	pD	DIC
9117.58	8987.00	130.59	9248.17

Here we see that the fit (D(thetabar)) is a lot better than the single level models that were fitted in Chapter 2 but that we now have 131 parameters and so the DIC value is only slightly better than the model with gender also included (of course we could fit gender here as well which would improve the fit further). The model we have just fitted is equivalent to fitting 65 separate regressions and so we are here considering each school as a completely separate entity.

In Chapter 3 we considered fitting the school effects as random terms and we can extend this idea by also considering the school×LRT effects as random. Here again we are using the idea that the schools are randomly chosen. We assume that schools are similar and so, as we have only taken a sample of pupils from each school, we wish to borrow strength from the other schools and shrink the LRT effects of each school towards the average LRT effect.

To fit the school effects as random we will first set up the variance components model from the last chapter. To do this we will clear our model and set up the variance components model from scratch (refer to Chapter 3 if you are unsure of how to do this).

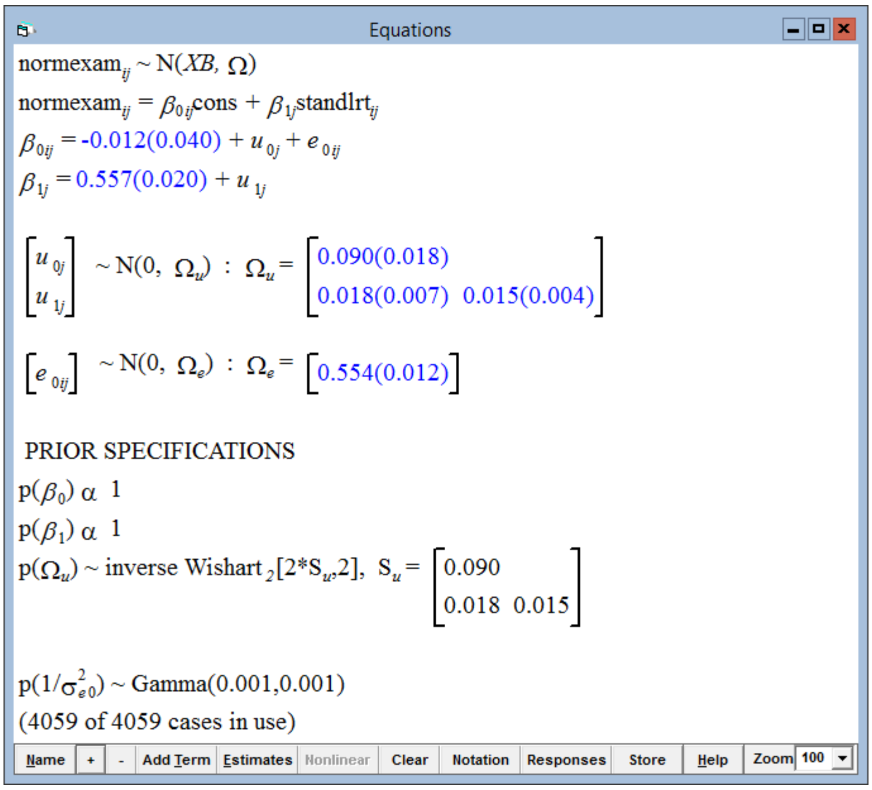
Next we need to add in the random effects for the **standlrt** (slope) variable:

- Click on **standlrt** in the **Equations** window.
- Click on the **j(school)** checkbox in the window that appears. This allows random LRT effects for each school.

The model we have now set up is often called a *random slopes regression model* as we can think of the LRT effects as slopes when we plot our predicted response variable against the LRT predictor for each school. We will first run the model using IGLS to obtain starting values before switching to MCMC:

- Click on the **Start** button.
- Click on the **MCMC** tab on the **Estimation Control** window.
- Click on the + button on the **Equations** window until prior distributions are shown.

After doing this the **Equations** window should look as follows:


Equations
Name + - Add Term Estimates Nonlinear Clear Notation Responses Store Help Zoom 100

$\text{normexam}_{ij} \sim \text{N}(XB, \Omega)$

$\text{normexam}_{ij} = \beta_{0ij}\text{cons} + \beta_{1j}\text{standlrt}_{ij}$

$\beta_{0ij} = -0.012(0.040) + u_{0j} + e_{0ij}$

$\beta_{1j} = 0.557(0.020) + u_{1j}$

$$\begin{bmatrix} u_{0j} \\ u_{1j} \end{bmatrix} \sim \text{N}(0, \Omega_u) : \Omega_u = \begin{bmatrix} 0.090(0.018) & \\ 0.018(0.007) & 0.015(0.004) \end{bmatrix}$$

$$\begin{bmatrix} e_{0ij} \end{bmatrix} \sim \text{N}(0, \Omega_e) : \Omega_e = \begin{bmatrix} 0.554(0.012) \end{bmatrix}$$

PRIOR SPECIFICATIONS

$p(\beta_0) \alpha\ 1$

$p(\beta_1) \alpha\ 1$

$$p(\Omega_u) \sim \text{inverse Wishart}_2[2*S_u, 2],\ S_u = \begin{bmatrix} 0.090 & \\ 0.018 & 0.015 \end{bmatrix}$$

$p(1/\sigma^2_{e0}) \sim \text{Gamma}(0.001, 0.001)$

(4059 of 4059 cases in use)

We can see that for a model that contains a variance matrix rather than a simple variance we use an inverse Wishart prior with as few degrees of freedom as possible. The Wishart prior family is not as convenient as the Gamma in that we have to include a prior guess for the variance matrix. Here we have a slightly data determined prior as by default MLwiN will take the current estimate of Ω_u (from the IGLS run) as a prior parameter. We will later compare this approach with some alternatives.

- Click the **Start** button to run the model.

The MCMC approach with default priors gives fairly similar estimates to the maximum likelihood approach. The DIC diagnostic can be calculated for this model and we find the following:

Dbar	D(thetabar)	pD	DIC
9122.99	9031.32	91.67	9214.65

In treating both the intercepts and slopes as random parameters we have reduced the effective number of parameters from 130.6 to 91.7. The fit of the random effects model is 44 points worse but the DIC diagnostic (accounting for the reduced number of parameters) suggests that this is a better model as its value has been reduced from 9248 to 9215.

6.1 Prediction intervals for a random slopes regression model

In Chapter 4 of the User's Guide to MLwiN details are given on how to use the **Predictions** window to construct predicted school lines with confidence intervals. Although we can construct a predicted line and intervals for the fixed part of the model easily using MCMC to include the school effects in the predictions we need to store the school level residuals as we require the MCMC chains of the predicted values to produce quantiles and hence give the prediction intervals.

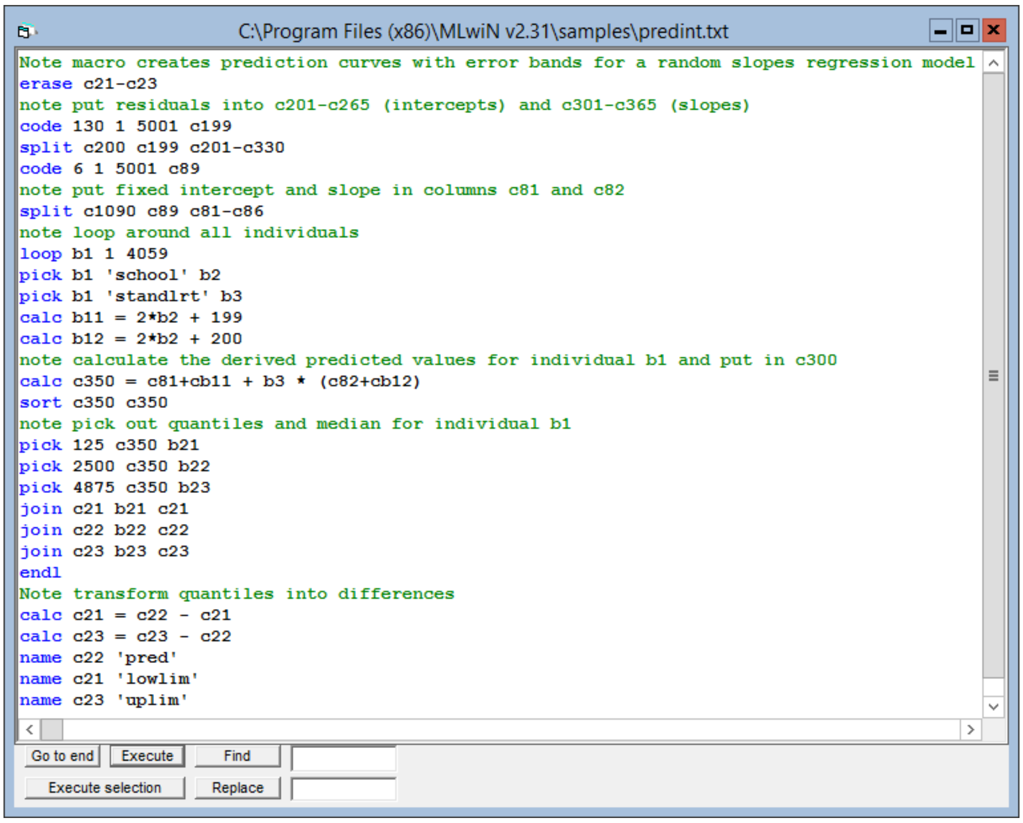
To set things up we will firstly need to run a model with the residuals stored as follows:

- Change estimation method to IGLS.
- Click on the **Start** button.
- Change estimation method to MCMC.
- On the **Estimation Control** window change the length of monitoring chain to 5001.
- Select **MCMC/Store Residuals** from the **Model** menu.
- Click in the box to **Store Level 2 Residuals**.
- Change **Column number** to **c200**.
- Click on the **Done** button.
- Click on the **Start** button.

This will run the random slopes model again, this time storing the residuals. We now need to create a point estimate (median) and interval estimate (quantiles) for the predicted value for each individual. To do this we have created a macro that generates the required quantities.

- Select **Open Macro** from the **File** menu.
- Select **predint.txt** from the list of files and click on the **Open** button.

The macro will now appear as shown below. This macro basically creates the chain of predicted values for each individual and takes the quantiles and median from this chain. Many of the commands are similar to those that we used in Chapter 4 to look at the chains for the residuals and the intra-school correlation.

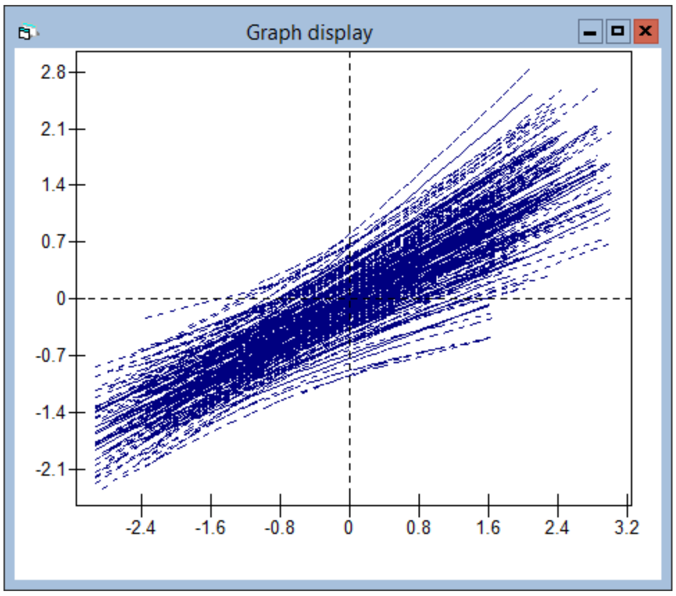


```
Note macro creates prediction curves with error bands for a random slopes regression model
erase c21-c23
note put residuals into c201-c265 (intercepts) and c301-c365 (slopes)
code 130 1 5001 c199
split c200 c199 c201-c330
code 6 1 5001 c89
note put fixed intercept and slope in columns c81 and c82
split c1090 c89 c81-c86
note loop around all individuals
loop b1 1 4059
pick b1 'school' b2
pick b1 'standlrt' b3
calc b11 = 2*b2 + 199
calc b12 = 2*b2 + 200
note calculate the derived predicted values for individual b1 and put in c300
calc c350 = c81+cb11 + b3 * (c82+cb12)
sort c350 c350
note pick out quantiles and median for individual b1
pick 125 c350 b21
pick 2500 c350 b22
pick 4875 c350 b23
join c21 b21 c21
join c22 b22 c22
join c23 b23 c23
endl
Note transform quantiles into differences
calc c21 = c22 - c21
calc c23 = c23 - c22
name c22 'pred'
name c21 'lowlim'
name c23 'uplim'
```

We can now run the macro by clicking on the **Execute** button. Note that this will take a couple of minutes. After the macro has finished to display our predictions we can use the **Customised graph** window.

- Select **Customised Graph** from the **Graph** menu.
- Select **pred** as the **y** variable.
- Select **standlrt** as the **x** variable.
- Select **school** as the **group** variable.
- Select **line** as **plot type**.
- Click on the **error bars** tab.
- Select **uplim** as the **y errors + variable**.
- Select **lowlim** as the **y errors − variable**.
- Select **lines** as the **y error type**.
- Click on the **Apply** button to plot the graph.

The graph will appear as follows:



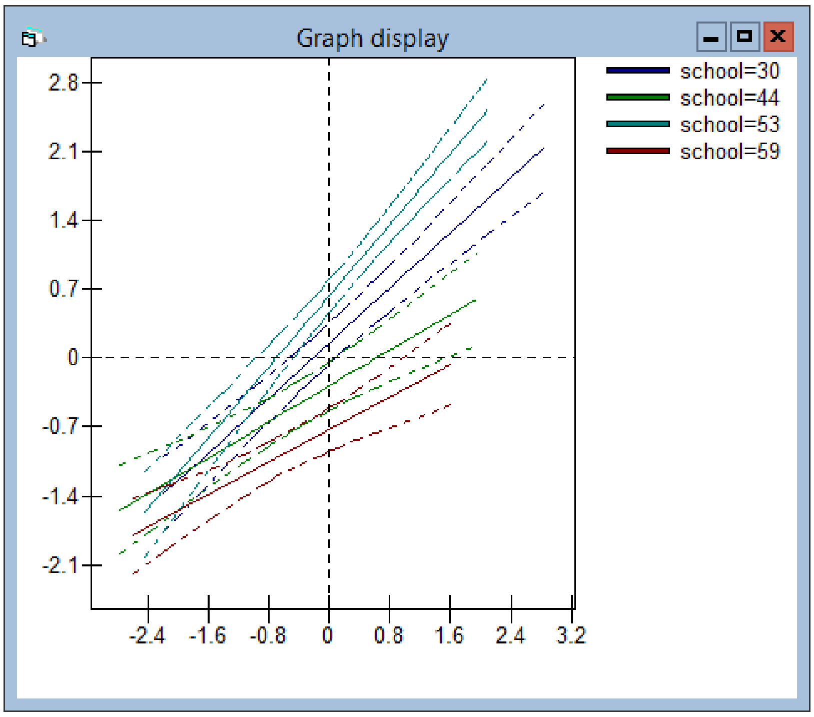
Here we see all 65 school lines plus their intervals on one graph. Due to the 195 lines on the one graph it is difficult to pick out the individual schools. We can however use a filter column to view just a few schools:

- Select **Command Interface** from the **Data Manipulation** menu.
- Type the following command to create the filter column:

```
▶ calc c24='school'==30 | 'school'==44 | 'school'==53
  | 'school'==59
```

- Select **Customised Graphs** from the **Graphs** menu.
- Select **c24** as the **filter variable**.
- Note if the **Customised Graphs** window is already visible you may need to close it and open it for **c24** to appear in the pull down list.
- Click on the **plot style** tab.
- Change **colour** to **16-rotate**.
- Click on the **other** tab.
- Click in the **group code** tickbox for key labels.
- Click on the **Apply** button.

The graphs for the four schools will then appear as follows:



So in this section we have shown that again when using MCMC methods, because we have draws from the joint posterior distribution of all unknown parameters, we can derive the distribution of any function of the parameters such as a prediction. Before continuing this chapter and looking again at prior distributions you should change the **monitoring chain length** back to 5,000 on the **Estimation Control** window.

6.2 Alternative priors for variance matrices

The default prior distribution that we have used in this example involves a slightly informative 'data-determined' prior for Ω_u. Browne (1998) and Browne & Draper (2000) perform some comparisons between some prior distributions that can be used as a default for a variance matrix. We will here, as a sensitivity analysis exercise, consider the effect of three alternative priors.

6.3 WinBUGS priors (Prior 2)

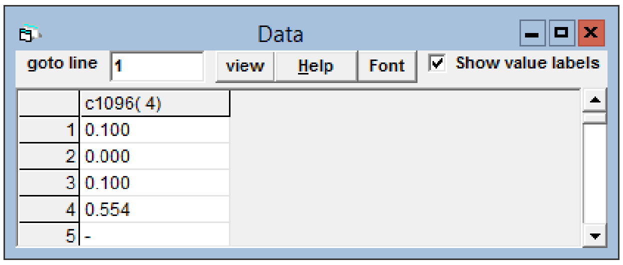
Spiegelhalter et al. (2000*b*) consider in their '*birats*' fitting a prior similar to our default, namely an inverse-Wishart with the smallest possible sample size and a prior guess that represents the magnitude of the variances. For example here we may use the values 0.1 for both the variances. To fit this prior we firstly need to:

- Run the model in IGLS

and then, as with when we changed the MCMC starting values, we can alter the parameter values in **c1096**:

- Select **View or Edit Data** from the **Data Manipulation** menu.
- Click on the **View** button.
- Select column **c1096** from the list.

This will bring up the data window and show the starting values for the variance parameters. We wish to change the first three values (level 2 variance). Type 0.1,0, and 0.1 in the first three rows and the window should look as follows:



	c1096(4)
1	0.100
2	0.000
3	0.100
4	0.554
5	-

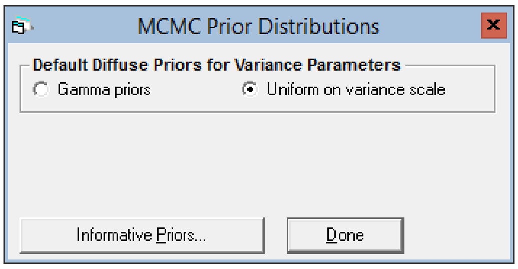
Now we have the priors set up.

- Select **MCMC** from the **Estimation** menu.
- Click on the **Start** button.

The results for this prior are given in the column headed Gibbs (prior 2) at the end of the chapter.

6.4 Uniform prior

Another alternative would be to fit a Uniform prior i.e. $p(\Omega_u) \propto 1$. To fit this prior we need to once again fit the model using IGLS. Then we need to change the default priors from gamma priors to Uniform on variance scale priors on the **MCMC priors** window (available from the **Model** menu) as shown below:


MCMC Prior Distributions
Default Diffuse Priors for Variance Parameters
Gamma priors
Uniform on variance scale
Informative Priors...
Done

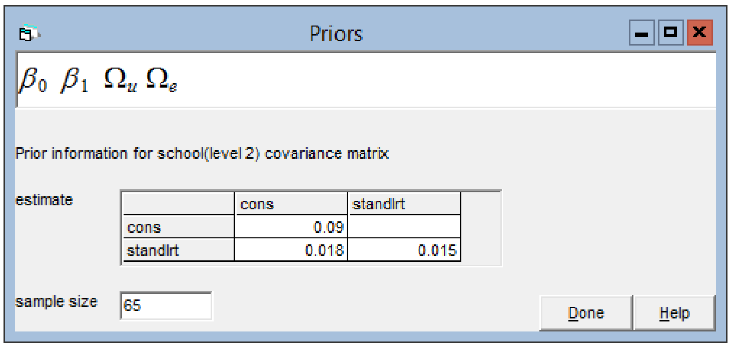
Note that this will change the prior for the level 1 variance as well. Fit this model and the results will be as in the column Gibbs (uniform) in the table at the end of the chapter.

6.5 Informative prior

Our final alternative is to assume (for illustration) that we had *a priori* collected another dataset with 65 schools and here the estimated variance matrix was identical to the IGLS estimate here. We then wish to use this as a prior distribution so (after as always running IGLS first) we:

- Select the **MCMC** tab from the **Estimation control** window.
- Select **MCMC/Priors** from the **Model** menu.
- Select **Gamma priors** from the **MCMC priors** window (this sets the prior for level 1 variance)
- Select **Informative Priors** from the **MCMC priors** window.
- Select Ω_u from the top bar of the window.
- Input the estimates **0.09**,**0.018** and **0.015** in the estimate boxes (as below).
- Input **65** in the sample size box.
- Click on Ω_u again.

When this is done the **Priors** window should look as follows:



Prior information for school(level 2) covariance matrix

estimate

	cons	standlrt
cons	0.09	
standlrt	0.018	0.015

sample size 65

- Click the **Start** button.

The results can be seen in the fourth column labelled Gibbs (prior 4) in the table below.

6.6 Results

The results for all 4 priors along with the IGLS estimates are given in the following table (Standard Errors in brackets). We can see that the results for all priors are similar which is reassuring. The second prior has an increased slopes variance as the prior guess was quite a bit higher than the IGLS estimate. The Uniform prior (as in Browne & Draper, 2000) is conservative and gives variance estimates that are biased high for all the variance parameters. The informative prior gives almost identical posterior estimates to its prior estimates, which is to be expected as we have given the prior equal weight to the data. We also see that the standard errors of the level 2 variances have reduced and so our estimates have greater precision. This is an advantage of the Bayesian approach in that when we have 'good' prior information we will get more precise estimates and hence smaller credible intervals.

Parameter	IGLS	Gibbs			
		default	prior 2	uniform	prior 4
β_0	-0.012 (0.040)	-0.006 (0.039)	-0.007 (0.039)	-0.006 (0.042)	-0.008 (0.042)
β_1	0.557 (0.020)	0.558 (0.020)	0.556 (0.023)	0.558 (0.022)	0.558 (0.020)
Ω_{u00}	0.090 (0.018)	0.096 (0.020)	0.096 (0.020)	0.103 (0.022)	0.091 (0.012)
Ω_{u01}	0.018 (0.007)	0.019 (0.007)	0.018 (0.008)	0.020 (0.008)	0.018 (0.004)
Ω_{u11}	0.015 (0.004)	0.015 (0.004)	0.023 (0.005)	0.018 (0.005)	0.015 (0.002)
σ^2_{e0}	0.554 (0.012)	0.554 (0.013)	0.553 (0.013)	0.554 (0.013)	0.554 (0.013)

Chapter learning outcomes

⋆ How to fit different LRT effects for each school in a fixed effects model.

⋆ How to fit a random slopes regression model.

- ⋆ How to compare models via the DIC diagnostic.
- ⋆ How to create prediction intervals for a model using MCMC.
- ⋆ A greater understanding of prior distributions.

Chapter 7

Using the WinBUGS Interface in MLwiN

We have so far looked at fitting Normal response models to continuous univariate data in MLwiN. We could consider fitting further models with additional fixed or random terms and these would simply be extensions to the models fitted thus far. There are, however, other extensions that we could consider; for example, heteroskedasticity of the response variable, which we consider in Chapter 9 and alternative distributions for the random effects that we will consider later in this chapter.

The MCMC features in MLwiN are fairly new and we currently fit only models of particular types although we are constantly extending the number of models that can be fitted. If, however, a user wishes to fit a model that cannot be currently fitted, for example fitting an alternative distribution for the school level random effects, there are three main options. Firstly wait for a later version of MLwiN that will fit their model; secondly write their own code to fit their model; or thirdly try an alternative software package, for example WinBUGS.

WinBUGS (Spiegelhalter et al., 2000*a*, freely available from `http://www.mrc-bsu.cam.ac.uk/bugs`) in its earlier guise of BUGS was one of the first Bayesian software packages and is a more general purpose Bayesian estimation engine than the MCMC engine in MLwiN. It works on a different philosophy of fitting models that can be represented by directed acyclic graphs (DAGs). BUGS has a compiled language which allows the user to specify their model through statements of two types—logical and distributional—which between them describe the structure of the DAG and hence the model. Then BUGS compiles this user code and constructs an MCMC estimation engine for the user's model that can be run to give chains of estimates in a similar way to the MLwiN engine.

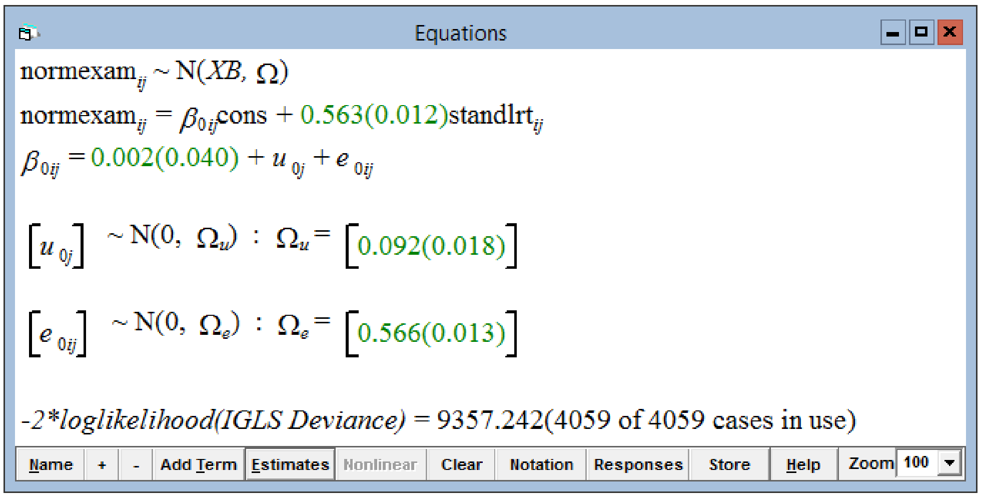
In this chapter we will firstly consider once again the variance components model and show how to fit this model in WinBUGS. We then go on to consider a model that MLwiN cannot fit which has t-distributed residuals at the school level. It should be noted at this point that most multilevel models are large and so cannot be run using the educational version of WinBUGS and so you will need to have the release version of WinBUGS.

7.1 Variance components models in WinBUGS

We will consider the tutorial dataset once again. Set up and run the variance components model with one explanatory variable (**standlrt**) using the IGLS method by

- Pressing the **Start** button

On IGLS convergence we get the following estimates:



$\text{normexam}_{ij} \sim \text{N}(XB, \Omega)$

$\text{normexam}_{ij} = \beta_{0ij}\text{cons} + 0.563(0.012)\text{standlrt}_{ij}$

$\beta_{0ij} = 0.002(0.040) + u_{0j} + e_{0ij}$

$\left[u_{0j}\right] \sim \text{N}(0, \Omega_u) : \Omega_u = \left[0.092(0.018)\right]$

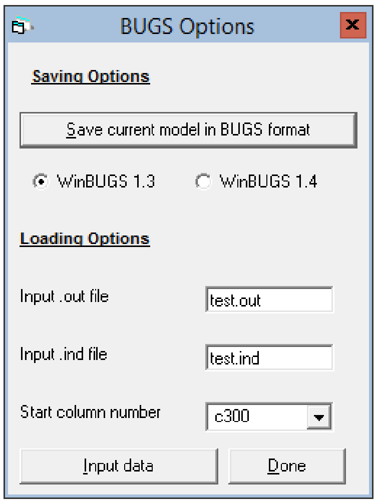
$\left[e_{0ij}\right] \sim \text{N}(0, \Omega_e) : \Omega_e = \left[0.566(0.013)\right]$

*-2*loglikelihood(IGLS Deviance)* = 9357.242(4059 of 4059 cases in use)

In Chapter 3 we then considered fitting this model using MCMC in MLwiN but here we will consider instead using WinBUGS. To get to the BUGS options in MLwiN we need to do the following:

- Click on the **Estimation Control** button.
- Select the **MCMC** tab.
- Select **MCMC/WinBUGS Options** from the **Model** menu.

This will bring up the **WinBUGS Options** screen that looks as follows:



Input .out file: test.out
Input .ind file: test.ind
Start column number: c300

From this screen we can save the BUGS code for the currently set up model or read in the output files that contain parameter traces from BUGS for use in MLwiN (see later). For now we will save our current model in BUGS format:

- Select the WinBUGS 1.4 button.
- Click on the large button at the top of the window.

This will bring up a file save window similar to those for inputting and saving worksheets. For now we will save the file in the default directory as **tutorial.bug**. This will create a file that contains the BUGS model definition, initial values and data. For users of classic BUGS who are used to having three separate files, in WinBUGS the file **tutorial.bug** contains the information from these three files with comment lines dividing them. In what follows we use WinBUGS version 1.4.3 and so it is possible that results will change with other versions.

For background information on using WinBUGS it is strongly suggested that the user reads some of the user manual and examples documentation that comes with the package, in particular to become familiar with the WinBUGS notation. For now to fit our model in WinBUGS, we must start the WinBUGS program and read in the file **tutorial.bug** (from the directory it was saved in) as a text file. Note that you will have to change the Files of type box to **All files (*.*)** to see the file **tutorial.bug**. Having read in the file a window headed **tutorial.bug** will appear containing the information needed by BUGS for this model.

As mentioned earlier the WinBUGS code is split into 3 sections and we will consider these here in turn. Firstly a model definition is required and this consists of a description of the structure of the current problem. The code for our simple variance components problem is as follows:

```
# WINBUGS 1.4 code generated from MLwiN program
```

```
#----MODEL Definition----------------

model
{
# Level 1 definition
for(i in 1:N) {
normexam[i] ~ dnorm(mu[i],tau)
mu[i]<- beta[1] * cons[i]
+ beta[2] * standlrt[i]
+ u2[school[i]] * cons[i]
}
# Higher level definitions
for (j in 1:n2) {
u2[j] ~ dnorm(0,tau.u2)
}
# Priors for fixed effects
for (k in 1:2) { beta[k] ~ dflat() }
# Priors for random terms
tau ~ dgamma(0.001000,0.001000)
sigma2 <- 1/tau
tau.u2 ~ dgamma(0.001000,0.001000)
sigma2.u2 <- 1/tau.u2
}
```

WinBUGS is a more general modelling package and so there is no standard order to the model description although when the code is generated from MLwiN it will generally have a similar structure. We firstly define the relationship between the response (in this example **normexam**) and the fixed and random predictor variables.

Note that the column names from MLwiN are used as the variable names in WinBUGS. WinBUGS does have some differences in what it allows as a variable name so if the WinBUGS code will not work it may be that some of your variable names are illegal, for example a column name like 1995 will be interpreted as a number in WinBUGS so it is worth renaming such columns in MLwiN.

So we see here that our response is normally distributed and that we have two fixed effects, **beta**[1] and **beta**[2] (always defined as **beta** by the code generator) and one set of random effects, **u2** (always defined as **u#** where # is the level/classification indicator). Note that in WinBUGS the fixed effects, **beta**, and all other vectors always start with index 1 and not 0 so that there will probably not be direct correspondence between the MLwiN and WinBUGS indexing. Next the code defines the random effects **u2** as being Normally distributed before finally giving the priors for the fixed effects and the variances.

WinBUGS has two types of relationship: distributional relationships that are described by the ∼ symbol and deterministic relationships that are described by the `<-` symbol which is also used in the S-plus package. Note that the normal distribution definition in WinBUGS, **dnorm**, has two parameters that are the mean and the precision (NOT the variance), hence the deterministic relationship used to calculate the variance. The prior distributions are identical to those used in the MCMC options in MLwiN.

Before running a model in WinBUGS we first need to read in the particular elements of the model using the **Specification** window available from the **Model** menu. After selecting the window containing the model by clicking on it, clicking on the **check model** button should give the message 'model is syntactically correct' at the bottom of the screen. Next we need to load in the data for the model. Due to the fact that the data is generally the largest part of the file generated by MLwiN it is included after the initial values. For the **tutorial.bug** example the data section begins as follows:

```
#----Data File---------------------------------

list(N= 4059, n2 = 65,
school = c(1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,
2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,3,3,3,3,3,3,
3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,
3,3,3,3,3,3,3,3,3,3,3,3,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4, ...
```

Again BUGS borrows its notation from S-plus using the convention c(...) to represent a vector of observations. Here we see the first two constants, **N** and **n2** that define the number of level 1 units and level 2 units, followed by a list of the **school** identifier for each observation. To load the data into BUGS we need to highlight the **list** identifier at the start of the data list and click on the **load data** button in the **specification** window. If this is successful the message 'data loaded' will appear at the bottom of the screen. Next we have to combine the data and model definition by clicking on the **compile** button. Again if this operation is successful a message appears at the bottom of the screen, this time stating that 'model compiled'.

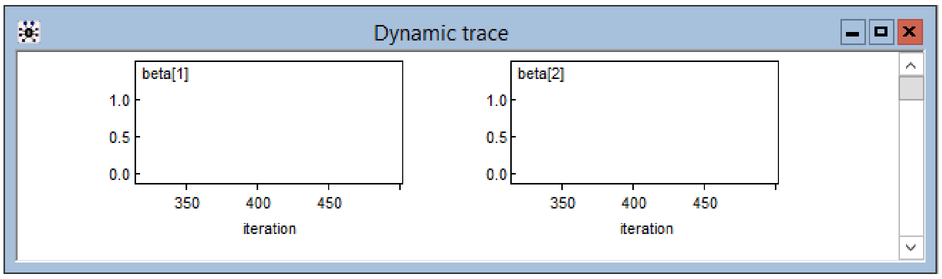
Finally as BUGS uses MCMC methods all unknown parameters will need starting values. These are included in the initial values part of the file that for our example is as follows:

```
#----Initial values file---------------------------

list(beta= c(0.002391,0.563371),
u2 = c( 0.373760,0.502043,0.503889,0.018131,0.240431,0.541395,
0.379002,-0.026173,-0.135181,-0.337021,0.179300,-0.061863,-0.149648,
-0.165592,-0.182922,-0.409984,-0.172780,-0.084464,-0.011510,0.214462,
0.244016,-0.435732,-0.489244,0.209408,-0.230472,-0.023543,0.023121,
-0.610002,0.240626,0.158475,0.033280,-0.006457,0.029590,-0.137882,
0.128634,-0.181341,-0.189077,-0.153068,0.130317,-0.234439,0.211543,
0.092820,-0.089927,-0.247556,-0.109729,-0.352727,-0.042628,-0.045058,
0.042845,-0.302412,-0.051373,0.381929,0.723314,-0.547252,0.503474,
0.009972,0.031894,0.138115,-0.658368,0.225656,-0.039551,-0.054029,
0.535641,0.087692,-0.165764),
tau= 1.767625,
tau.u2= 10.854523)
```

This gives the estimates from the IGLS run for the fixed effects and precisions, and an MLwiN **RESI** command for the initial values for **u2** that are exactly what the MCMC routine in MLwiN uses as starting values. To use these values in WinBUGS we need to highlight the **list** identifier at the start of the initial values and click on the **load inits** button on the **specification** window. This will then give the final message 'initial values loaded; model initialized'. Note that WinBUGS will generate starting values for any parameters that have not explicitly been given starting values but here we have given all parameters starting values.

We are now ready to run the Gibbs sampler in the WinBUGS package. Before we start the estimation engine we have to tell WinBUGS which parameters we wish to monitor. We will choose the same parameters as MLwiN uses. From the **Inference** menu select the **Samples** options and a window will appear that allows the user to specify which parameters to monitor. In this window we will firstly select the fixed effects by typing **beta** in the **node** box. Note that when a correctly typed parameter is input the **set** button will become enabled. We will also want to use a burn-in of 500 iterations so we will also modify the beg value from 1 to 501. After this press the **set** button and the parameter will be set for monitoring. We now need to repeat this procedure with the two variance parameters **sigma2** and **sigma2.u2**.

It should be noted that it is possible in WinBUGS to get dynamic traces of the parameters like those in the **Trajectories** window in MLwiN via the **sample** window. If we either type **beta** again in the **node** box or use the scroll button at the side of the box to select **beta** you will see that now all the buttons become enabled. Clicking the **trace** button will give 2 empty trace plots for **beta**[1] and **beta**[2] (as shown below), which will become dynamic when we start updating. Similar traces can be brought up for the two variance parameters.



We are now ready to set the estimation engine running and this is done via the **Update** window found in the **Model** menu. We need to specify the number of updates (including the burn-in) and so we will replace the 1000 here with 5500 as is used in MLwiN. As in MLwiN you can also specify how often to refresh the screen, whether to use thinning and, if WinBUGS needs to use Metropolis Hastings sampling, whether to adapt. There is also the option to use a technique called over-relaxation that improves the mixing of the MCMC chains but takes longer per iteration. The current version of WinBUGS (1.3) will choose the MCMC routines it uses for you depending on the form of the conditional distribution (see section 1.3 of the WinBUGS manual for details).

Now that we have set the number of iterations press the **update** button to start the sampler. After a few minutes (depending on the speed of your processor and how many traces you are viewing) the update counter will reach 5500 and the sampling will be finished. WinBUGS has the nice feature that it will give you a message at the bottom of the screen, for example 'updates took 88s', stating how long the sampling took which is useful for comparing model run times etc. Generally WinBUGS is slower for models that can also be run in MLwiN but as we will see later it has greater flexibility in the models it can fit and sometimes the MCMC methods it uses are more efficient than the Metropolis Hastings methods used in MLwiN for binomial and Poisson response models.

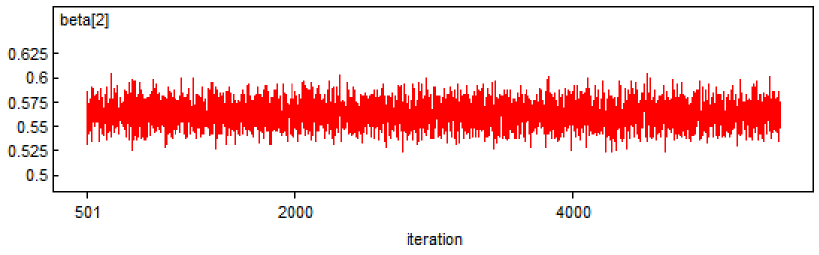
Once the sampling has finished we can now look at the estimates, plots and other information again via the **sample** window. To get summary information, select **beta** in the **node** box and click on the **stats** button. A **node statistics** window will appear giving the following

node	mean	sd	MC error	2.5%	median	97.5%	start	sample
beta[1]	0.002979	0.03995	0.002516	-0.07465	0.004435	0.07912	501	5000
beta[2]	0.5634	0.01264	1.997E-4	0.5388	0.5636	0.5882	501	5000

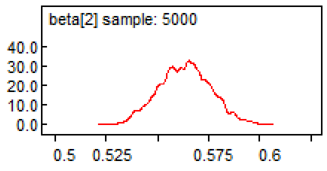
These results are similar to those obtained from MLwiN, and we can also get similar results for the other parameters as shown below.

node	mean	sd	MC error	2.5%	median	97.5%	start	sample
sigma2	0.5661	0.0127	1.653E-4	0.5416	0.5662	0.5905	501	5000
sigma2.u2	0.09662	0.02019	3.256E-4	0.06415	0.09446	0.1426	501	5000

We can also get trace plots and kernel density plots via the **history** and **density** buttons respectively. Below we see the trace plot for the parameter beta[2]



and the kernel plot.

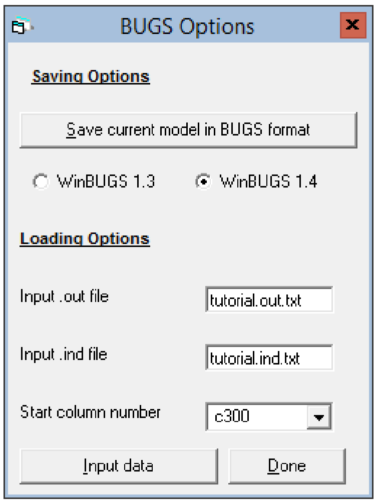


Currently WinBUGS does not allow the smoothing parameter to be changed for the kernel plots so that they look rather crude but this will be changed in later versions. Note here that we could have typed * in the node box to get statistics or plots for all monitored nodes.

WinBUGS currently produces limited summary statistics and plots itself. Historically the plots and MCMC diagnostics were provided via a suite of S-plus functions called CODA (Best et al., 1995), and WinBUGS also has the option to produce the input files that CODA requires. MLwiN can also use these files to input the parameter chains from WinBUGS into columns in MLwiN.

Here we will consider all parameters by using the * option so select this in the node box and press the **coda** button on the **sample** window. This will produce two windows that are labelled **CODA index** which contains the variable names and **CODA for chain 1** which contains the values for the parameter chains. We will now save these files as text files by clicking on the respective windows and then choosing **Save As** from the **File** menu. We will need to save the files in *plain text* (*.txt) format. We will store the **CODA index** file as **tutorial.ind** and the **CODA for chain 1** file as **tutorial.out** in the same directory as **tutorial.bug**. Note that these are the extensions that the classic BUGS used for these files but, as we have selected the plain text format, WinBUGS will add an additional .txt to the index filename and so the files are actually saved as **tutorial.ind.txt** and **tutorial.out.txt**.

Now back in MLwiN if you want to input the traces return to the **BUGS options** window that we used earlier (available from the **Model** menu).



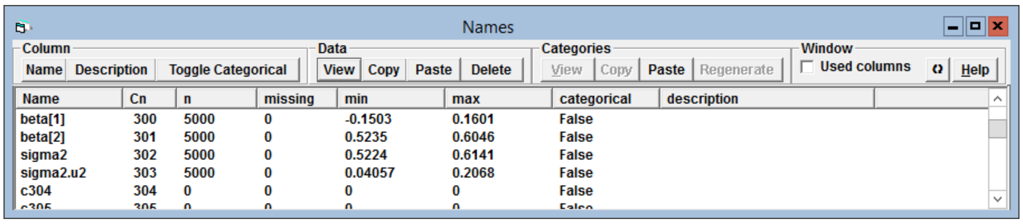
○ WinBUGS 1.3 [✓ WinBUGS 1.4]

Input .out file: tutorial.out.txt

Input .ind file: tutorial.ind.txt

Start column number: c300

Here we will need to modify the **.out** and **.in** file fields to **tutorial.out** and **tutorial.ind.txt** respectively. Note that if you did not put these files in the current directory you will have to include their full path names in the respective boxes. The window should then look as above. Pressing the **Input data** button will now load the chains into columns **c300** to **c304**. To confirm this bring up the **Names** window from the **Data Manipulation** window and scroll down to **c300** and you will see the following:

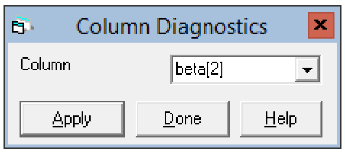


Name	Cn	n	missing	min	max	categorical	description
beta[1]	300	5000	0	-0.1503	0.1601	False	
beta[2]	301	5000	0	0.5235	0.6046	False	
sigma2	302	5000	0	0.5224	0.6141	False	
sigma2.u2	303	5000	0	0.04057	0.2068	False	
c304	304	0	0	0	0	False	

We can now use the MLwiN MCMC diagnostics on the BUGS output, for example for the slope parameter:

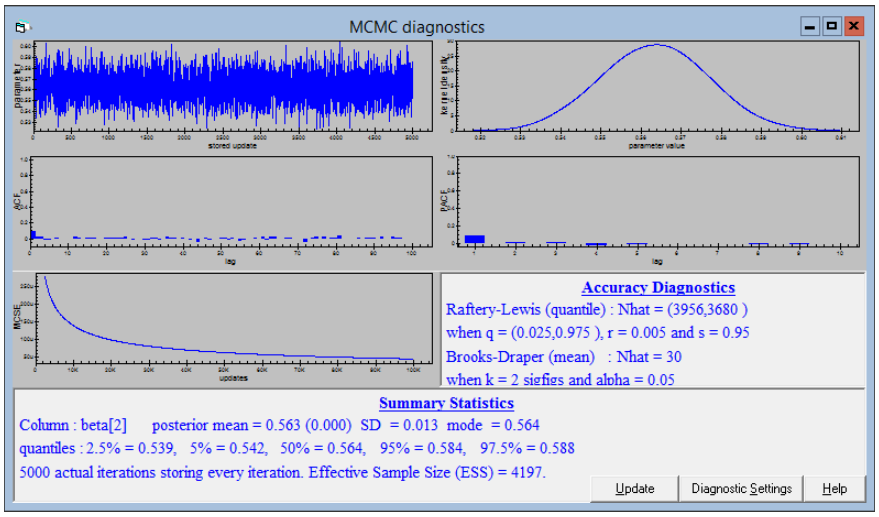
- Select the **Column Diagnostics** window from the **Basic Statistics** window.
- Select the column labelled **beta[2]**.

The window should then look as follows:



Column: beta[2]

Note that this parameter is the fixed effect for the slope that is labelled β_1 in MLwiN. Clicking on the **Apply** button will give the following diagnostics screen, which is very similar to that given by the MLwiN MCMC sampler in earlier chapters.



Accuracy Diagnostics
Raftery-Lewis (quantile) : Nhat = (3956,3680)
when q = (0.025,0.975), r = 0.005 and s = 0.95
Brooks-Draper (mean) : Nhat = 30
when k = 2 sigfigs and alpha = 0.05

Summary Statistics
Column : beta[2] posterior mean = 0.563 (0.000) SD = 0.013 mode = 0.564
quantiles : 2.5% = 0.539, 5% = 0.542, 50% = 0.564, 95% = 0.584, 97.5% = 0.588
5000 actual iterations storing every iteration. Effective Sample Size (ESS) = 4197.

We can repeat all of the above procedures for the intercept parameter and the two variances and we will see that we get similar results for all four parameters with both MLwiN and WinBUGS.

7.2 So why have a WinBUGS interface ?

The example we have just gone through will give similar results using both software packages and to use WinBUGS we have to move back and forth between the two packages. Also the estimation engine in WinBUGS is slower, so you may be asking yourself the above question. The interface was written originally as a testing tool to confirm that when new types of models are programmed into MLwiN we get the same answers as WinBUGS. We recommend that you check that both packages give similar answers.

7.3 t distributed school residuals

The main advantage of having a WinBUGS interface however, is to allow models that have not yet been developed in MLwiN to be fitted using WinBUGS. We will illustrate this by considering alternative distributions for the school level residuals in the tutorial example we considered earlier. In the User's Guide to MLwiN we look at plots of residuals against normal scores to confirm that the normal distributional assumption is a good fit to the data.

The normal distribution is a member of the t distribution family. The t distribution family has an additional degrees of freedom (df) parameter and the normal distribution is the limiting case when this parameter reaches infinity. We will here consider replacing the normal distribution at level 2 with a t distribution where the **df** parameter has itself got a prior distribution. For this we will use a uniform prior and allow the **df** parameter to take values in the range 1 to 200. This allows both small values, where the distribution has very long tails, and large values, which are indistinguishable from the normal distribution.

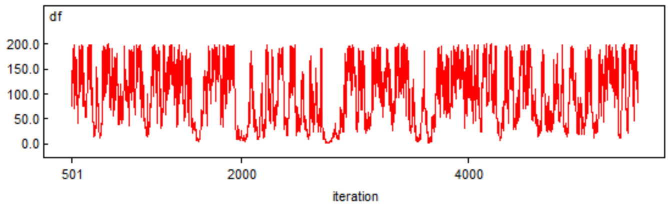
To include this prior we will need to edit the model definition in the file **tutorial.bug**. The new version is as follows (edits in bold font):

```
#----MODEL Definition----------------

model
{
# Level 1 definition
for(i in 1:N) {
normexam[i] ~ dnorm(mu[i],tau)
mu[i]<- beta[1] * cons[i]
+ beta[2] * standlrt[i]
+ u2[school[i]] * cons[i]
}
# Higher level definitions
for (j in 1:n2) {
u2[j] ~ dt(0,tau.u2,df)
}
# Priors for fixed effects
for (k in 1:2) { beta[k] ~ dflat() }
# Priors for random terms
tau ~ dgamma(0.001,0.001)
sigma2 <- 1/tau
tau.u2 ~ dgamma(0.001,0.001)
sigma2.u2 <- 1/tau.u2
df ~ dunif(2,200)
}
```

We will also need to give a starting value for **df** in the initial values file and so we will choose (arbitrarily) **df** = 10. Our initial values file then looks as follows:

```
#----Initial values file----------------------------

list(beta= c(0.002391,0.563371),
u2 = c( 0.373760,0.502043,0.503889,0.018131,0.240431,0.541395,
0.379002,-0.026173,-0.135181,-0.337021,0.179300,-0.061863,-0.149648,
-0.165592,-0.182922,-0.409984,-0.172780,-0.084464,-0.011510,0.214462,
0.244016,-0.435732,-0.489244,0.209408,-0.230472,-0.023543,0.023121,
-0.610002,0.240626,0.158475,0.033280,-0.006457,0.029590,-0.137882,
0.128634,-0.181341,-0.189077,-0.153068,0.130317,-0.234439,0.211543,
0.092820,-0.089927,-0.247556,-0.109729,-0.352727,-0.042628,-0.045058,
0.042845,-0.302412,-0.051373,0.381929,0.723314,-0.547252,0.503474,
0.009972,0.031894,0.138115,-0.658368,0.225656,-0.039551,-0.054029,
0.535641,0.087692,-0.165764),
tau= 1.767625,
tau.u2= 10.854523,
```
df = 10)

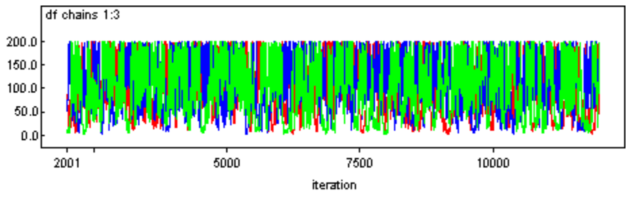
This time we will monitor the same four parameters as before plus the df parameter which we will **set** in the **sample** window. Note that the **adapting** box on the **update** window is ticked because for this model, WinBUGS 1.4 uses a method called slice sampling to update the **df** parameter. Note that the tick disappears when adapting has finished. We again run for 5000 iterations after a burn-in of 500 iterations and get the following trace for df:



Here we see reasonably good mixing. Earlier versions of WinBUGS (1.3) didn't use the slice sampler and then this parameter did not mix but the slice sampler has improved on this. We can see from the summary statistics below that on this small sample of 5000 iterations we cannot reject the possibility of a heavy-tailed distribution.

node	mean	sd	MC error	2.5%	median	97.5%	start	sample
df	97.83	57.75	4.409	8.13	94.96	194.5	501	5000

In order to investigate the potential of starting value dependence we started three chains with identical parameter starting values except for *df*, which was set to 2, 10 and 200 respectively for the 3 runs. To do this in WinBUGS is fairly easy as on the specification window there is a **num of chains** box that we edit to 3 (immediately after checking the model is syntactically correct). Then load the data and compile before loading the 3 sets of initial values. This simply involves editing the `df=10` line of the initial value file before loading each set. We will increase the burn-in to 2000 by changing the **beg** box to 2001 on the **sample** window when we input the parameters we wish to monitor. We will also increase the updates to 12000 so that we have a monitoring run of 10000 iterations after the burn-in. Pressing **update**, the three sets of chains will be run concurrently and so this will take longer.



If we look at plots of the 3 sets we see that, all 3 chains are mixing well and there is strong overlap suggesting that sensitivity to the starting value is not a problem. If we look at the summary statistics for the three chains combined we get:

node	mean	sd	MC error	2.5%	median	97.5%	start	sample
df	101.5	56.12	1.832	9.671	101.2	195.1	2001	30000

This summary information suggests that a very small degrees of freedom (**df**) parameter and hence an extremely heavy tailed distribution is not likely but that a value of **df** of less than 10 is not out of the question.

As a sensitivity analysis we will instead try fitting a model where the degrees of freedom is assumed known and has value 8 which suggests a slightly long-tailed distribution at level 2. Seltzer (1993) gives Gibbs sampling algorithms for exactly this scenario of a known df parameter. We will need to simplify our model definition as follows:

```
# WINBUGS 1.4 code generated from MLwiN program

#----MODEL Definition----------------

model
{
# Level 1 definition
for(i in 1:N) {
normexam[i] ~ dnorm(mu[i],tau)
mu[i]<- beta[1] * cons[i]
+ beta[2] * standlrt[i]
+ u2[school[i]] * cons[i]
}
# Higher level definitions
for (j in 1:n2) {
u2[j] ~ dt(0,tau.u2,df)
}
# Priors for fixed effects
for (k in 1:2) { beta[k] ~ dflat() }
# Priors for random terms
tau ~ dgamma(0.001,0.001)
sigma2 <- 1/tau
tau.u2 ~ dgamma(0.001,0.001)
sigma2.u2 <- 1/tau.u2
df <- 8
}
```

and we will also need to remove the initial values for parameter **df** as it is now a constant. Again we will monitor **beta, sigma2** and **sigma2.u2**. After running for 5000 iterations after a burn-in of 500 we get the following estimates:

node	mean	sd	MC error	2.5%	median	97.5%	start	sample
beta[1]	-0.001065	0.04125	0.002364	-0.08322	-7.715E-4	0.07792	501	5000
beta[2]	0.5633	0.0124	1.979E-4	0.5391	0.5636	0.5872	501	5000
sigma2.u2	0.07657	0.01805	4.247E-4	0.04746	0.0745	0.1171	501	5000
sigma2	0.5662	0.01279	1.787E-4	0.5421	0.5659	0.5916	501	5000

Here we see that the fixed effects and level 1 variance are little changed in terms of point estimate and standard errors, suggesting the analysis is robust to different level 2 distributions. The level 2 variance parameter is not directly comparable as the variance of the t distribution is a function of both the **sigma2.u2** and **df** parameters.

In fact the variance is $8/6 \times 0.07657 = 0.102$, which is slightly higher than for the Normal case.

We will investigate the WinBUGS interface further when we consider binary response models in Chapter 10 and for several examples in later chapters. Here we will look at how the different procedures used in WinBUGS for these models compare with the methods used in MLwiN.

Chapter learning outcomes

- ⋆ How to create WinBUGS code from the MLwiN package.
- ⋆ How to run models in WinBUGS.
- ⋆ How to output chains from WinBUGS back into MLwiN.
- ⋆ How to fit t distributed residuals in WinBUGS.
- ⋆ How to check the sensitivity of the Gaussian assumption at level 2 of the model.

Chapter 8

Running a Simulation Study in MLwiN

This book describes how to fit various statistical models using the MCMC methodology available in MLwiN. One of the main questions people have when faced with a new methodology is what is its advantage over the current method I am using? An alternative question that is often faced when using MCMC methods is 'Which prior should I use for my model?' When we wish to compare which method or which 'default' priors are 'best' for our particular model and dataset we are faced by the problem of not knowing what the correct estimates should be. There are also issues about in what sense is a method 'better' than another method and depending on your criterion you may get different conclusions.

One of the best ways to compare different estimation methods is to run a simulation study. Here we generate simulated datasets where the true values of the parameters are known and so we have a 'gold standard' to compare our estimates with. MLwiN is especially suited for running simulation studies as the macro language underlying the package can be used both to generate simulated datasets and fit models via several estimation methods. In this chapter we demonstrate how to perform one particular simulation study, and give macro code that can then be altered by the reader to fit alternative simulation studies.

8.1 JSP dataset simulation study

Browne (1998) performed several simulation studies to compare likelihood-based methods with Bayesian MCMC methods with several alternative 'default' prior distributions. These simulations were extended in Browne & Draper (2000, 2006). Browne used as the basis for his simulations a small educational dataset from the Junior School Project (JSP) (Mortimore et al.,

1988) and created many datasets with the same or similar structure to this actual dataset. We will here consider one of the smallest models that Browne looked at which contained 108 pupils spread evenly over 6 schools. In our example we will simply compare the IGLS maximum likelihood method with the MCMC method with default ($\Gamma(\varepsilon, \varepsilon)$) priors.

To run a simulation study consists of four basic steps, two of which are repeated:

1. Set up the structure of the dataset

Repeat the next 2 steps N times.

2. Generate a simulated dataset based on the true parameter values
3. Fit the model to the simulated dataset using all the methods to be compared.
4. Analyse the results of the N simulations.

We will now deal with the four steps in turn.

8.2 Setting up the structure of the dataset

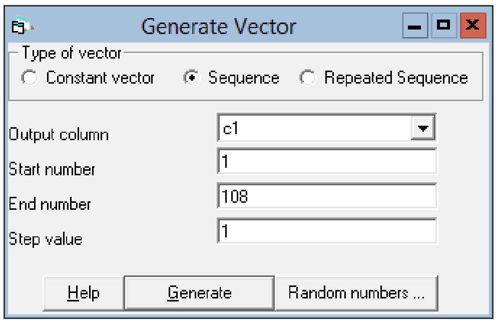
For the purposes of our simulation we are going to fit a variance components model with no predictors to a dataset with 108 pupils in 6 schools. Consequently before we start we need to create 3 columns: a pupil id, a school id and a constant vector for the intercept term. We will also need to create a response variable although this will be generated for each simulated dataset so, for now, we will create a dummy constant response.

All of these columns can be created by the **Generate Vector** window but as we wish to run our simulations in batch mode we will need to write macro commands instead. Fortunately virtually all the window buttons that perform an action in MLwiN have an associated command in the command language. For more details on the command language see the Command manual or the online help where information on a particular command can be found in the index under ‘Command XXXX’ where XXXX is the name of the command.

So to start our macro we will do the following:

- Start up MLwiN.
- Select the **Generate vector** window from the **Data Manipulation** menu.
- Select **Sequence**.
- Select **Output column c1**.
- Select **Start number** 1.
- Select **End number** 108.
- Select **Step value** 1.

The window should now look as follows:



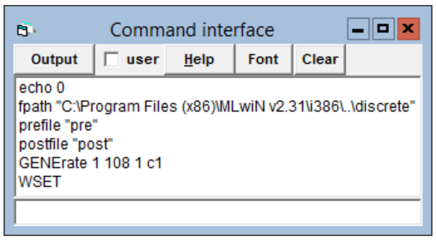
Type of vector: ( ) Constant vector [✓ Sequence] ( ) Repeated Sequence

Output column: c1
Start number: 1
End number: 108
Step value: 1

Now click on **Generate** and column **c1** will contain the numbers 1–108 to represent the pupil identifiers. If we now look at the **Command interface** window:

- Select the **Command Interface** window from the **Data Manipulation** menu.
- Remove the tick by the **user box** by clicking on it.

You should now see a lot of commands that MLwiN has performed when starting up, and the window should look something like the following:

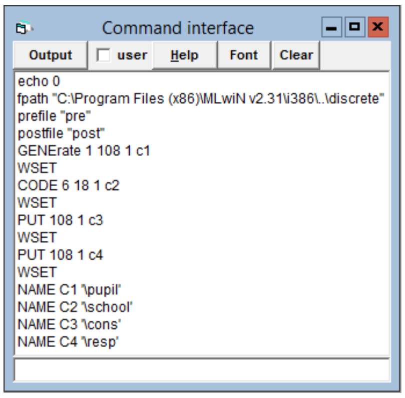


```
echo 0
fpath "C:\Program Files (x86)\MLwiN v2.31\i386\..\discrete"
prefile "pre"
postfile "post"
GENErate 1 108 1 c1
WSET
```

The important command here is the **GENErate** command that creates the vector of level 1 identifiers. We now also need to create level 2 identifiers and a constant vector for both the intercept and the response. This can be done using the **Generate Vector** window as follows:

- Select **Repeated Sequence**.
- Select **Output column c2**.
- Select **Maximum Number** 6.
- Select **Number of Repeats per Block** 18.
- Select **Number of Blocks** 1.
- Select **Generate**.
- Select **Constant Vector**.
- Select **Output Column c3**.
- Select **Number of Copies** 108.
- Select **Value** 1.
- Select **Generate**.
- Select **Output Column c4**.
- Select **Generate**.

This will set up the four columns **c1**–**c4** and we can then name these columns using the **Names** window. We will name the columns **pupil**, **school**, **cons** and **resp**. After naming these four columns the **Command interface** window will look as follows:



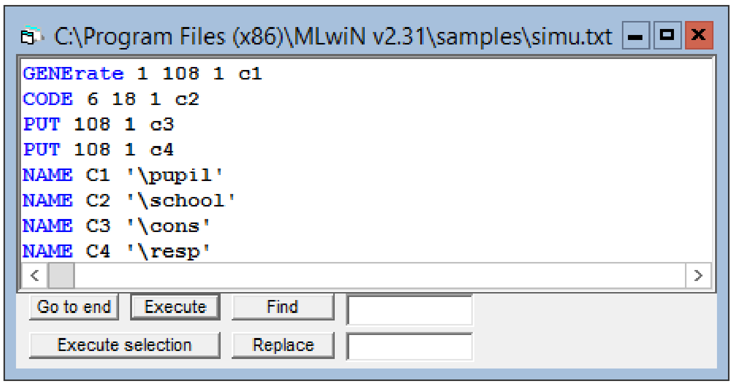
```
echo 0
fpath "C:\Program Files (x86)\MLwiN v2.31\i386\..\discrete"
prefile "pre"
postfile "post"
GENErate 1 108 1 c1
WSET
CODE 6 18 1 c2
WSET
PUT 108 1 c3
WSET
PUT 108 1 c4
WSET
NAME C1 '\pupil'
NAME C2 '\school'
NAME C3 '\cons'
NAME C4 '\resp'
```

So here we see that setting up the four columns and naming them uses 8 commands. (We can ignore the WSET commands). We could copy these commands into a macro so that rather than typing these commands we could instead just execute the macro. To do this:

- Select **New Macro** from the **File** Menu.
- Copy the 8 lines into this new macro window (highlight them using the mouse then press Ctrl+C, move to the Macro window and press Ctrl+V).
- Select **Save Macro As** from the **File** Menu.

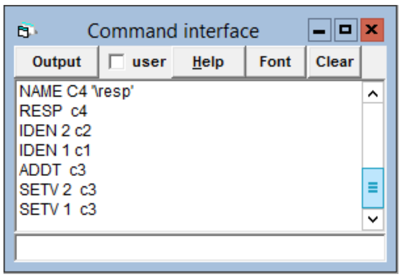
- Browse to find the folder where the sample worksheets are located (somewhere like ‘C:\Program Files (x86)\MLwiN v2.31\samples’)
- Type the file name **simu** into the **Save** window.
- Click on the **Save** button.

The macro window should now look as follows (note the directory name may be different on your machine):



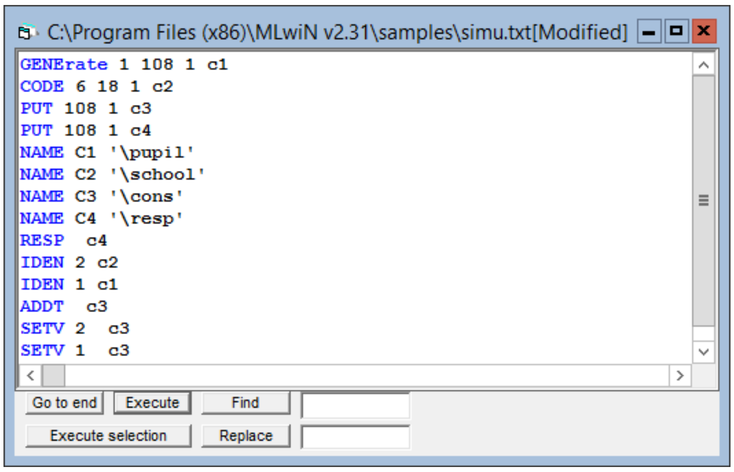
```
GENErate 1 108 1 c1
CODE 6 18 1 c2
PUT 108 1 c3
PUT 108 1 c4
NAME C1 '\pupil'
NAME C2 '\school'
NAME C3 '\cons'
NAME C4 '\resp'
```

Note that any commands the macro window recognizes as valid macro commands it will colour blue. We now need to set up the variance components model as usual. Former users of MLN will know that there are also commands that can be used to set up models rather than the **Equations** window. For now set up a model in the **Equations** window that has **resp** as the response, 2 levels with **school** as level 2 and **pupil** as level 1 and one predictor **cons** which is a fixed effect and random at both levels 1 and 2. If you are not sure how to do this then you should re-read Chapter 2. Viewing the **Command interface** window we now see the following:



```
NAME C4 '\resp'
RESP  c4
IDEN 2 c2
IDEN 1 c1
ADDT  c3
SETV 2  c3
SETV 1  c3
```

So here we have 6 commands that have set up the model. We can now add these 6 commands to our macro so that we now have:


C:\Program Files (x86)\MLwiN v2.31\samples\simu.txt[Modified]

```
GENErate 1 108 1 c1
CODE 6 18 1 c2
PUT 108 1 c3
PUT 108 1 c4
NAME C1 '\pupil'
NAME C2 '\school'
NAME C3 '\cons'
NAME C4 '\resp'
RESP  c4
IDEN 2 c2
IDEN 1 c1
ADDT  c3
SETV 2  c3
SETV 1  c3
```

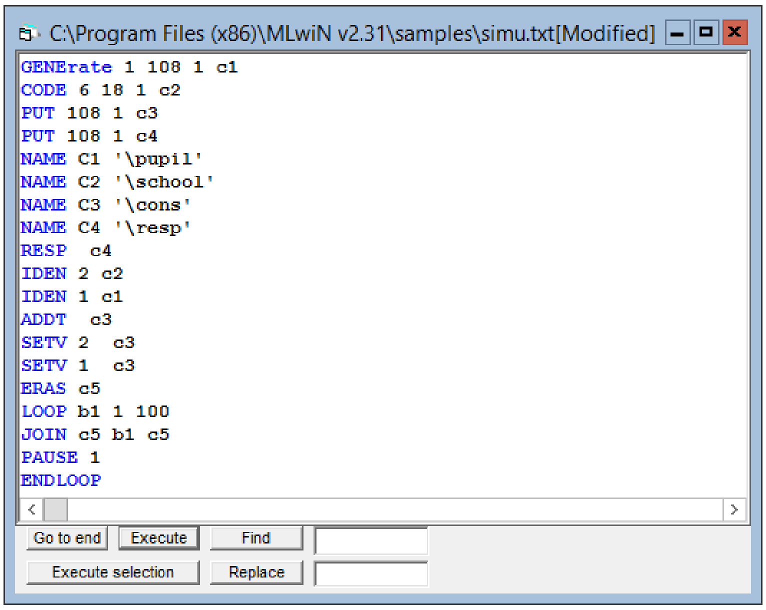
We have not yet saved our macro, which we can tell from the '[Modified]' that appears on the top bar of the window, so now save the macro again by selecting **Save Macro** from the **File** menu. We can now exit MLwiN and restart. To get back to our position we simply need to:

- Select **Open Macro** from the **File** menu.
- Select **simu**.
- Click on **Open**.
- On the macro window that appears click on **Execute**.

(Note that in this case a quicker way to open the macro is to select it from the list of recently used macros at the bottom of the File menu.) So we have a macro that will set up the basic model structure. Now we will extend this macro so that it generates simulated datasets.

8.3 Generating simulated datasets based on true values

In order to perform repeated actions we now need to use several macro commands that are not commonly used when running MLwiN in interactive mode. These commands are **LOOP**, **ENDLOOP**, **OBEY**, **PAUSE**, **SEED** and **JOIN**. We will have already seen most of these commands in the macros we looked at in Chapter 1. The **LOOP** and **ENDLOOP** commands allow us to repeat a series of actions several times and the **JOIN** command will allow us to join the results of each action onto the end of a column so that later we can store the results of all our simulations in one column. The **PAUSE 1** command as mentioned in Chapter 1 is an escape command in that it momentarily stops the macro at a point and updates all the windows. We can illustrate these commands in action by modifying our macro to include the following extra commands:


C:\Program Files (x86)\MLwiN v2.31\samples\simu.txt[Modified]
Go to end
Execute
Find
Execute selection
Replace

```
GENErate 1 108 1 c1
CODE 6 18 1 c2
PUT 108 1 c3
PUT 108 1 c4
NAME C1 '\pupil'
NAME C2 '\school'
NAME C3 '\cons'
NAME C4 '\resp'
RESP  c4
IDEN 2 c2
IDEN 1 c1
ADDT  c3
SETV 2  c3
SETV 1  c3
ERAS c5
LOOP b1 1 100
JOIN c5 b1 c5
PAUSE 1
ENDLOOP
```

Before clicking on the **Execute** button you should now do the following:

- Select **View/Edit Data** from the **Data Manipulation** menu.
- Click on the **View** button.
- Select **c5**.
- Click on **OK**.
- On the **Macro** window click on the **Execute** button.

This macro, as should be evident when you click on **Execute**, simply writes the loop number (stored in box b1) to the end of the column c5 and refreshes the screen after each number. Interesting as this is, we would actually prefer to do something more useful, at each iteration, than simply write out the iteration number. As we can see our macro has now become quite long. In macro writing as with other forms of programming it is useful to introduce structure into the code to aid readability. Although the current version of the Macro language does not include the concept of a function (that takes arguments) we can use the **OBEY** command to redirect a macro to another macro which can be thought of as a function with no arguments.

Here we will firstly write a short macro that will generate a simulated response based on the true settings for all the parameters. For brevity for the rest of this chapter we will simply list commands rather than go through the whole process of describing exactly how to perform the same actions via the menus and windows. To aid in your understanding you may want to try and perform the same actions using the windows to confirm you get the same answers.

Note that MLwiN reserves the columns **c1096** and **c1098** for storing the parameter estimates for the current model. To generate a response we therefore need to set the three values in **c1096** and **c1098** to the true answers for the variance parameters and the fixed effects respectively. This is performed via the **EDIT** command, which is equivalent to editing the numbers in the **Data** window. For example the command

```
▶ EDIT 1 c1096 10
```

will put the value 10 in the first position of **c1096** which corresponds to the level 2 variance estimate. For the level 1 variance we use the true value 40 and for the intercept we use the value 30.

Now given the true values we need to generate a random response from the random part of the model and add this to the fixed part of the model. Here we use the following command

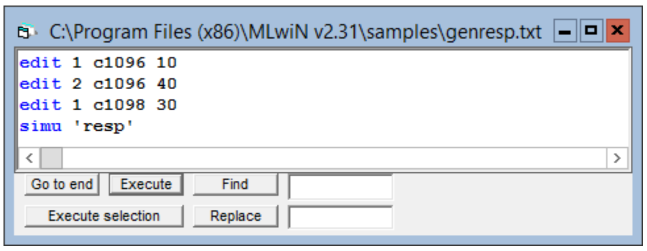
```
▶ SIMU 'resp'
```

This is a special command that takes the fixed effects and variance estimates, generates residuals at each level and by adding these to the fixed effects constructs a random response vector.

So to include these four commands in a macro do the following:

- Select **New Macro** from the **File** menu.
- Type the four commands into the macro.
- Save the macro as **genresp**.

The macro should look as follows:

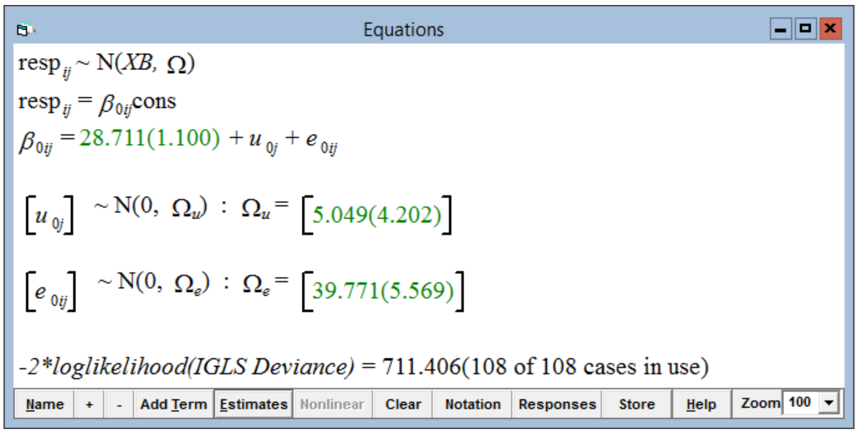


```
edit 1 c1096 10
edit 2 c1096 40
edit 1 c1098 30
simu 'resp'
```

We can now execute this macro and then run the model.

- Click on the **Execute** button on the macro window.
- Click on the **Start** button.

As we have not changed the estimation settings MLwiN will have run the model using our generated response using the IGLS method. If we look at the estimates in the **Equations** window we should have something like the following:



$\text{resp}_{ij} \sim \text{N}(XB, \Omega)$

$\text{resp}_{ij} = \beta_{0ij}\text{cons}$

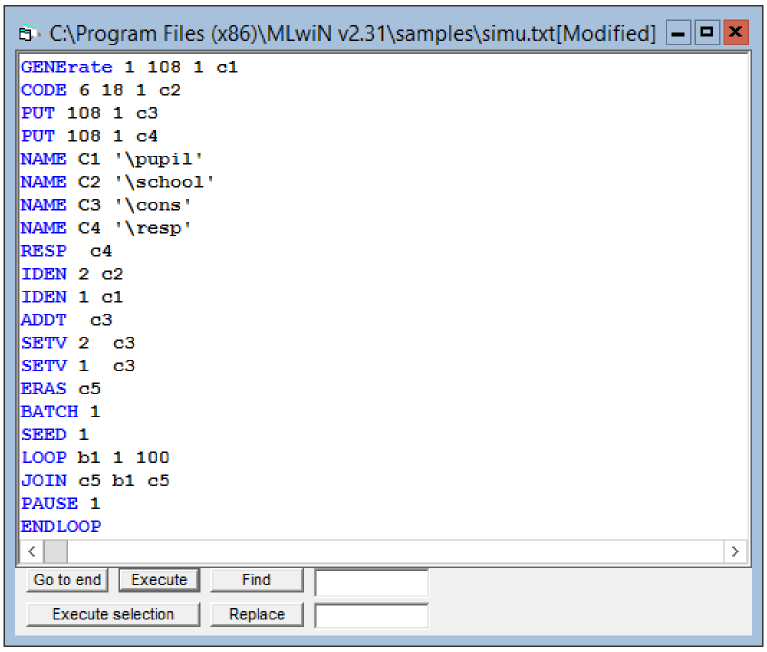
$\beta_{0ij} = 28.711(1.100) + u_{0j} + e_{0ij}$

$\left[u_{0j}\right] \sim \text{N}(0,\ \Omega_u) \ : \ \Omega_u = \left[5.049(4.202)\right]$

$\left[e_{0ij}\right] \sim \text{N}(0,\ \Omega_e) \ : \ \Omega_e = \left[39.771(5.569)\right]$

*-2*loglikelihood(IGLS Deviance)* = 711.406(108 of 108 cases in use)

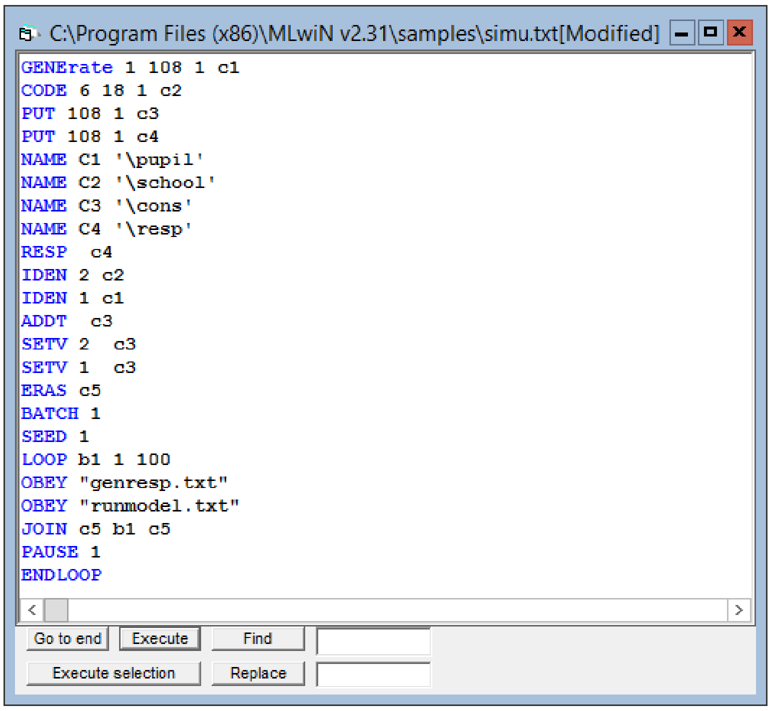
Note that we have not yet added a **SEED** command so you may get different estimates due to having a different random response vector. We now need to link up our macro for generating random responses via the **OBEY** command. You now need to alter the **simu** macro so that it looks as follows:


C:\Program Files (x86)\MLwiN v2.31\samples\simu.txt[Modified]
Go to end
Execute
Find
Execute selection
Replace

```
GENErate 1 108 1 c1
CODE 6 18 1 c2
PUT 108 1 c3
PUT 108 1 c4
NAME C1 '\pupil'
NAME C2 '\school'
NAME C3 '\cons'
NAME C4 '\resp'
RESP  c4
IDEN 2 c2
IDEN 1 c1
ADDT  c3
SETV 2  c3
SETV 1  c3
ERAS c5
BATCH 1
SEED 1
LOOP b1 1 100
JOIN c5 b1 c5
PAUSE 1
ENDLOOP
```

Here we have added two commands: **SEED 1** that sets the random number seed before the simulations begin and **BATCH 1** that tells MLwiN we are running the macro in batch mode and so when we run the IGLS method (see later) the macro will run to convergence rather than just for 1 iteration. Note that the **OBEY** command includes the path name and so you may need to modify this on your machine.

So we now have a macro that when run will generate 100 response vectors, we now need to add a macro that will run the methods and store the results in columns. This macro we will call **runmodel** and all results will be output to columns **c11–c15** so we will erase these columns before we start. We therefore need to make the following final modifications to macro **simu**:


C:\Program Files (x86)\MLwiN v2.31\samples\simu.txt[Modified]
Go to end
Execute
Find
Execute selection
Replace

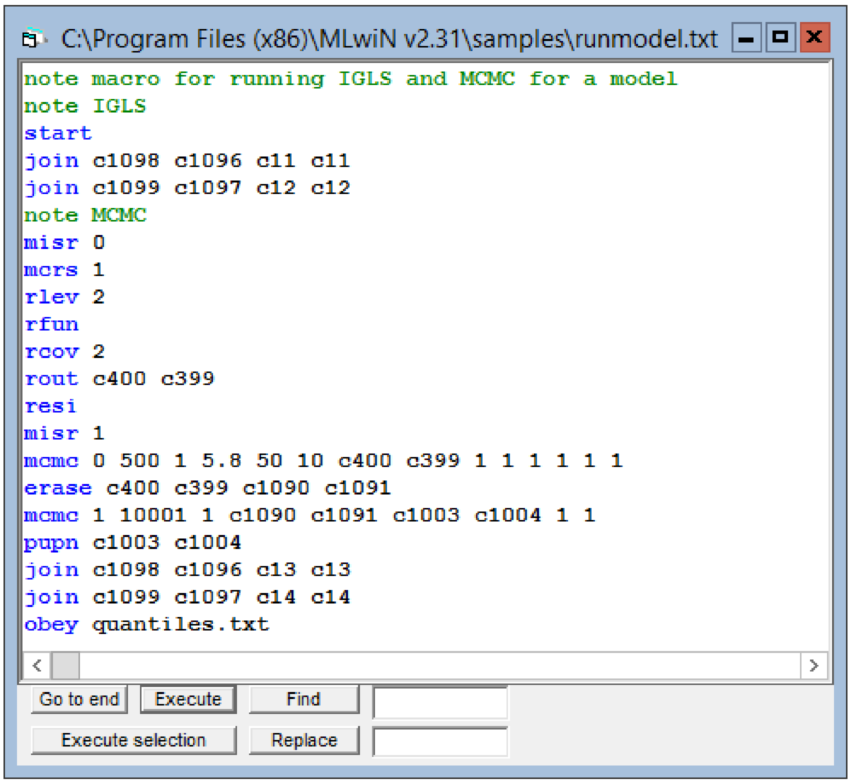
```
GENErate 1 108 1 c1
CODE 6 18 1 c2
PUT 108 1 c3
PUT 108 1 c4
NAME C1 '\pupil'
NAME C2 '\school'
NAME C3 '\cons'
NAME C4 '\resp'
RESP  c4
IDEN 2 c2
IDEN 1 c1
ADDT  c3
SETV 2  c3
SETV 1  c3
ERAS c5
BATCH 1
SEED 1
LOOP b1 1 100
OBEY "genresp.txt"
OBEY "runmodel.txt"
JOIN c5 b1 c5
PAUSE 1
ENDLOOP
```

8.4 Fitting the model to the simulated datasets

We now need to write a macro that will run both the IGLS method and the MCMC method. To discover the commands that IGLS and MCMC uses you can run both methods and look at the commands output although there will be many commands, in particular for MCMC. This is because the MCMC command is called each time the screen is refreshed and also the software uses the IGLS estimates and the **RESI** command to set up good starting values for the residuals. To save you typing the commands yourself the macro can be loaded:

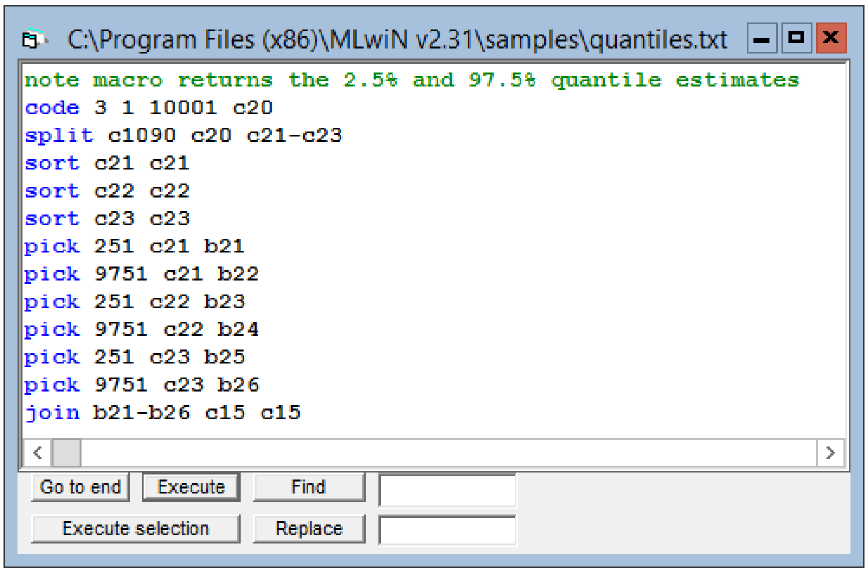
- Select **Open Macro** from the **File** menu.
- Select **runmodel** from the list of files and click on **Open**.

The macro opened should look as follows:



```
note macro for running IGLS and MCMC for a model
note IGLS
start
join c1098 c1096 c11 c11
join c1099 c1097 c12 c12
note MCMC
misr 0
mcrs 1
rlev 2
rfun
rcov 2
rout c400 c399
resi
misr 1
mcmc 0 500 1 5.8 50 10 c400 c399 1 1 1 1 1 1
erase c400 c399 c1090 c1091
mcmc 1 10001 1 c1090 c1091 c1003 c1004 1 1
pupn c1003 c1004
join c1098 c1096 c13 c13
join c1099 c1097 c14 c14
obey quantiles.txt
```

Here we first run the IGLS method and store the estimates in **c11** and their variances in **c12**. Then we run the MCMC method using the residuals and estimates from IGLS as starting values and run for 10,001 iterations after a burnin of 500. Note in their simulations Browne & Draper ran for 50,000 iterations, and so it is easy to change the macro to do this, although here for speed we use a shorter main run. We store the estimates in **c13** and their variances in **c14**. For the Bayesian 95% credible intervals we need to calculate the quantiles of the posterior distributions from the chains of values and for this we have another short macro as shown below:



```
note macro returns the 2.5% and 97.5% quantile estimates
code 3 1 10001 c20
split c1090 c20 c21-c23
sort c21 c21
sort c22 c22
sort c23 c23
pick 251 c21 b21
pick 9751 c21 b22
pick 251 c22 b23
pick 9751 c22 b24
pick 251 c23 b25
pick 9751 c23 b26
join b21-b26 c15 c15
```

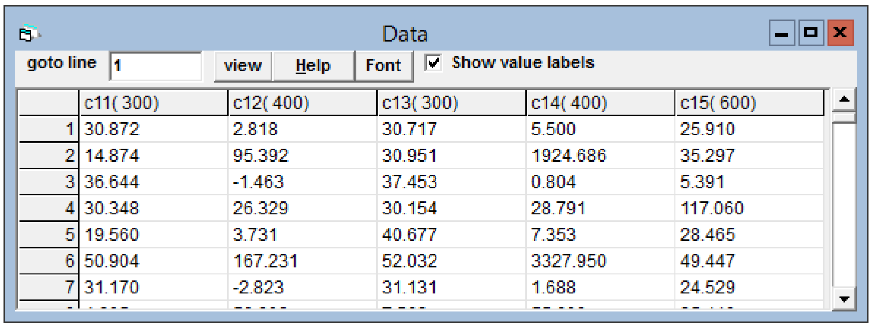
To calculate the 2.5% percentile point we need to find the value where 2.5%

of the other values in the chain are lower and 97.5% are higher hence we use a chain of 10,001 iterations rather than 10,000 for ease of calculation. We firstly separate the three chains using the **SPLIt** command and then sort each of them and pick out the correct values storing them stacked in column **c15**.

We can now run our first macro **simu.txt** to perform our simulation. In order to watch the progress it will be worth opening the **Data** window and viewing the columns where the output will appear. Note that we have used the **JOIN** command in such a way that the estimates for each dataset will appear at the top of each column. Note also that if you wish to view the output it is important to ensure that the **Macro** window is not above the **Data** window. Finally, note that because we have used commands like 'OBEY genresp.txt' we will need to change the current directory to the directory where these files are located. (We could alternatively have used the full pathname instead of just genresp etc. and then we would not have needed to change directory).

- Select **View/Edit Data** from the **Data Manipulation** menu.
- Select columns **c11**–**c15** from the view option of this window.
- Stretch the **Data** window so all five columns are shown (they will be currently empty).
- Select **dIrectories** from the **Options** menu.
- Click the **Browse** button next to the **current directory** box, locate the folder where you saved the macros and click **OK**.
- Click **Done**.
- Click on the **Execute** button on the **simu.txt** macro window.

Running the 100 simulations may take a little time. For each simulation the column **c11** will have three estimates added to it and so as an indicator of progress this column will be of length 300 when the simulations are complete. Upon the macro finishing the window should look as follows:



goto line 1
[✓ Show value labels]

	c11(300)	c12(400)	c13(300)	c14(400)	c15(600)
1	30.872	2.818	30.717	5.500	25.910
2	14.874	95.392	30.951	1924.686	35.297
3	36.644	-1.463	37.453	0.804	5.391
4	30.348	26.329	30.154	28.791	117.060
5	19.560	3.731	40.677	7.353	28.465
6	50.904	167.231	52.032	3327.950	49.447
7	31.170	-2.823	31.131	1.688	24.529

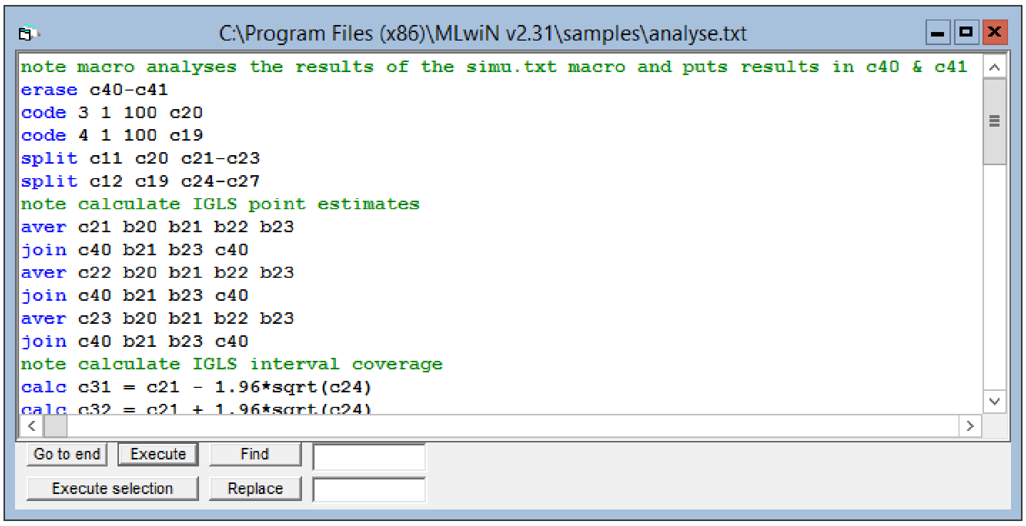
All five output columns contain information for the 100 simulations stacked up.

Here **c11** and **c13** have the three sets of estimates stored for the IGLS and

MCMC methods respectively. Columns **c12** and **c14** contain the variances of the three estimates plus the covariances between the two variance estimates. Column **c15** contains the MCMC 95% credible interval end points for the 3 parameters. We now need to transform these five columns into some sensible summary measures.

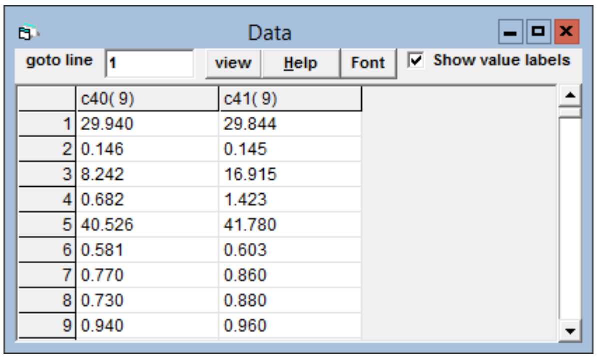
8.5 Analysing the simulation results

In order to summarise the results of 100 simulations we will follow the example of Browne & Draper (2006) and consider the bias and interval coverage properties of the two methods. For this we have written another macro **analyse.txt**. This macro, which is shown in part below, involves finding, as point estimates, the average of the 100 simulations for each method. For interval estimates, for the IGLS method we need to calculate the endpoints of 95% confidence intervals, and for this we use central Gaussian (mean ± 1.96×sd) intervals. See Browne & Draper (2006) for information on alternative intervals to be used with the IGLS method. The MCMC credible intervals have already been calculated. Next we have the simple task of counting the number of simulations where the true value is between the interval end points for each interval.



```
note macro analyses the results of the simu.txt macro and puts results in c40 & c41
erase c40-c41
code 3 1 100 c20
code 4 1 100 c19
split c11 c20 c21-c23
split c12 c19 c24-c27
note calculate IGLS point estimates
aver c21 b20 b21 b22 b23
join c40 b21 b23 c40
aver c22 b20 b21 b22 b23
join c40 b21 b23 c40
aver c23 b20 b21 b22 b23
join c40 b21 b23 c40
note calculate IGLS interval coverage
calc c31 = c21 - 1.96*sqrt(c24)
calc c32 = c21 + 1.96*sqrt(c24)
```

The above macro involves separating the output columns via the **SPLIt** command, and then calculating the required summary statistics via the **CALC** and **AVER** commands. The results are put in **c40** for IGLS and **c41** for the MCMC method as shown below:



	c40(9)	c41(9)
1	29.940	29.844
2	0.146	0.145
3	8.242	16.915
4	0.682	1.423
5	40.526	41.780
6	0.581	0.603
7	0.770	0.860
8	0.730	0.880
9	0.940	0.960

These results show the point estimates with standard errors for the three parameters and the percentage of intervals that cover the true parameter value for each parameter.

These results are similar to those in Browne & Draper (2006). Both methods show very little bias for the intercept (true value = 30) and the MCMC procedure only has a slight bias for the level 1 variance (true value = 40, MCMC mean estimate = 41.78). The level 2 variance however has both methods giving biased estimates. The true value is 10 and IGLS gives an average estimate of 8.25 (a 17.5% negative bias) whilst MCMC gives an average estimate of 16.91 (a 69.1% positive bias). As Browne & Draper show the median estimate using the $\Gamma(\varepsilon, \varepsilon)$ prior has much better bias properties (a 0.6% negative bias in their simulations) and this macro could be easily modified to also calculate the medians for each simulation.

In terms of interval coverage the MCMC method gives much better coverage for both the intercept and the level 2 variance. The level 1 variance has good coverage under both methods. Note however that if we were to look at the interval widths we will find, like Browne & Draper, that the MCMC methods have much larger intervals. Of course running only 100 simulations is not really enough to compare the methods properly and more simulations would be preferred. It would be easy to alter the above macros to run for 1000 simulations with the MCMC method main run length increased to 50,000 like Browne & Draper but this will take longer to run. It would also be easy to monitor the median for the MCMC method and the interval widths for both methods.

The simulations in Browne & Draper originally took a few months to run but given the advances in speed of processors running ALL the simulations in their paper would today be much quicker.

Chapter learning outcomes

⋆ How to run simulation experiments in MLwiN.

- ⋆ How to use many new Macro commands.
- ⋆ How to compare estimation methods for a particular model.

Chapter 9

Modelling Complex Variance at Level 1 / Heteroscedasticity

One level normal response models assume that all responses are (conditional on predictor values) independent observations from a Gaussian distribution with an unknown mean structure and constant variance. The interest in fitting one level models such as regressions and other linear models lies in improving the description (in terms of fit to existing data and predictive power) of the unknown mean function. In this book so far we have extended these one level models to the multilevel modelling framework while maintaining a constant level 1 variance.

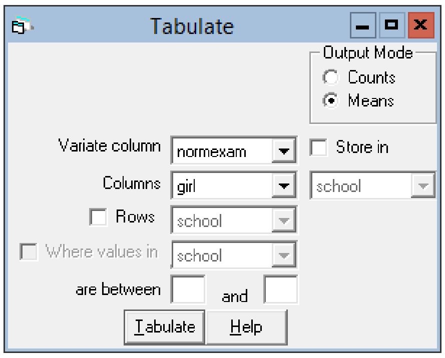
The primary goal of multilevel modelling is to adjust inferences on parameter estimates to account for non-independence between observations. The variance components models described in Chapters 3–5 do this by adjusting for correlation between responses taken from the same 'higher level unit'. This means that we assume that two responses taken from the same 'higher level unit', for example school, are more likely to be similar than two responses chosen at random.

Although variance components models adjust for correlation between responses in a cluster they still assume a constant variance across responses, which they split into components at the various levels of the model. This constant variance assumption may not be true and the variance of the responses may, like the mean of the responses, be a function of predictor variables. When the variance is treated as a function of the predictor variables this is known as 'heteroscedasticity' or complex variation. In this chapter we will look at fitting models that account for this complex variation using MCMC methods. Browne et al. (2002) consider exactly this problem and give further details about the exact algorithms that we use later.

To illustrate 'heteroscedasticity' we can consider the **tutorial** dataset again. Before fitting any models to the dataset we could consider the values of the response for different subsets (partitions) of the dataset. This partitioning of the dataset may then suggest predictors to fit in the model. To look at partitions of the dataset we can use the **Tabulate** window in MLwiN:

- Select **Tabulate** from the **Basic Statistics** menu.
- Select **Means** as the **Output Mode**.
- Select **normexam** for the **Variate column** from the pull down list.
- Select **girl** from the pull down list to the right of **Columns**.

The screen should then look as follows:



Output Mode: Means
Variate column: normexam
Columns: girl
Rows: school
Where values in: school
are between and

- Press the **Tabulate** button.

	0	1	TOTALS
N	1623	2436	4059
MEANS	-0.140	0.0933	-0.000114
SDs	1.03	0.970	0.992

Here we have split the dataset into boys (0) and girls (1). It is noticeable that the two subsets differ both in terms of mean and standard deviation and hence variance (although the difference in standard deviation is pretty small). We saw in Chapter 2 that girls did significantly better than boys but there we assumed a constant variance for the two groups. In Chapter 2 our main predictor of interest is the intake score, **standlrt** (**c5**). This variable is continuous so in order to tabulate the response we will need to create partitions of the dataset based on ranges of intake variable. Here we aim to split the dataset into seven roughly equal partitions. To do this we create a column, named **intakecat**, which labels the partitions 0 (lowest intake scores) to 6 (highest intake scores). Enter the following commands at the **Command Interface** window:

```
▶ Calc c11 = (c5>(-1)) + (c5>(-0.5)) + (c5>(-0.1)) +
  (c5>0.3) + (c5>0.7) + (c5>1.1)
▶ Name c11 'intakecat'
```

Now if we return to the **Tabulate** window we can display the means for these seven partitions as follows:

- Select **intakecat** from the pulldown list to the right of **Columns**.
- Select **Means** as the **Output Mode**.
- Press the **Tabulate** button.

The output for this tabulate command is as follows:

	0	1	2	3	4	5	6	TOTALS
N	612	594	619	710	547	428	549	4059
MEANS	-0.887	-0.499	-0.191	0.0439	0.278	0.571	0.963	-0.000114
SDs	0.855	0.774	0.806	0.811	0.812	0.824	0.838	0.817

Here, as in Browne et al. (2002), the partitions 0 to 6 refer to intake scores in the ranges less than -1, -1 to -0.5, -0.5 to -0.1, -0.1 to 0.3, 0.3 to 0.7, 0.7 to 1.1, and greater than 1.1 respectively. We can see that the means for the partitions increase with intake score and this corresponds to the positive coefficient found for the intake fixed effect in Chapter 1. We can, however, also see that the standard deviation (and therefore the variance) of the partitions is not constant and is also, with the exception of the first category, increasing. This pattern with larger variances at both extreme categories suggests that perhaps assuming a quadratic relationship between intake score and the variance would be sensible. We will next explain the modifications required to the MCMC algorithms to fit non-constant variance functions before returning to the **tutorial** example. Readers not interested in the algorithmic details can skip the next section whilst those who wish to know more can read Browne et al. (2002).

9.1 MCMC algorithm for a 1 level Normal model with complex variation

We will here consider fitting the linear regression model from Chapters 1 and 3 but with the additional assumption that the variance is a quadratic function of the reading test predictor. The model can be written

$$\begin{aligned} y_i &= \beta_0 + \beta_1 x_{1i} + e_i \\ e_i &\sim N(0, \sigma^2_{ei}) \qquad \text{where } \sigma^2_{ei} = X^T_{ei}\Omega_e X_{ei} = \Omega_{e0,0} + 2x_{1i}\Omega_{e0,1} + x^2_{1i}\Omega_{e1,1} \end{aligned}$$

Here although we think of the variance as a quadratic function, we maintain the matrix formulation Ω_e used by the IGLS method. The vector X_{ei} contains the predictors used in the level 1 variance for individual i, in this case the constant predictor ($x_0 = 1$ for all individuals) and the intake score (x_1). Note that if we wish to fit a simpler linear variance function we can constrain $\Omega_{e1,1}$ to equal zero.

To fit the model in a Bayesian framework we need to add prior distributions for the unknown parameters, β_0, β_1, and Ω_e. We will use the default priors from MLwiN, $p(\beta_0) \propto 1$, $p(\beta_1) \propto 1$ and $p(\Omega_{ej,k}) \propto 1$ for all j, k subject to the constraint that $\sigma^2_{ei} > 0$ for all i. This prior is equivalent to a uniform prior on all matrices Ω_e that satisfy the constraint and is the only prior available for these parameters in MLwiN.

The algorithm now consists of two steps. Firstly the fixed effects vector $\beta = (\beta_0, \beta_1)^T$ is updated using Gibbs sampling from its bi-variate Normal conditional posterior distribution:

$$p(\beta|y, \sigma^2_{ei}) \sim N_2(\hat{\beta}, \hat{D}), \qquad \text{where}$$
$$\hat{\beta} = \hat{D}\left[\sum_i \frac{X^T_{ei} y_i}{\sigma^2_{ei}}\right], \quad \text{and} \quad \hat{D} = \sum_i \frac{\sigma^2_{ei}}{X^T_i X_i}$$

The second step then involves updating the three parameters that make up the level 1 variance matrix Ω_e. We cannot update these terms using Gibbs sampling and so instead we will use Metropolis Hastings sampling. We will update each parameter in turn and will describe here the step to update $\Omega_{e0,1}$ as the other steps are similar. As in the Metropolis macro in Chapter 1 we need to use a proposal distribution but this time we will use a truncated Normal proposal in order to satisfy the constraints that the variance function must be positive for all observations.

We can write for every observation i,

$$\sigma^2_{ei} = 2x_{1i}\Omega_{e0,1} - \delta_{i0,1}, \quad \text{where} \quad \delta_{i0,1} = -\Omega_{e0,0} - x^2_{1i}\Omega_{e1,1}$$

Now for the condition $\sigma^2_{ei} > 0 \ \forall i$ to hold after we update $\Omega_{e0,1}$ we need to calculate the truncation points where the condition ceases to hold. This will result, in this case, in two sets of constraints that produce the two truncation points for our proposal distribution (note that there will be only one truncation point for the other two parameters as they are multiplied by terms that are strictly positive). The constraints can be written as follows:

$$m_u = \min_i \left(\frac{\delta_{i0,1}}{2x_{1i}} \forall i, x_{1i} < 0 \right) > \Omega_{e0,1} > \max_i \left(\frac{\delta_{i0,1}}{2x_{1i}} \forall i, x_{1i} > 0 \right)$$

So the Metropolis Hastings step at iteration t generates a proposed value $\Omega^*_{e0,1}$ from a Normal distribution with mean $\Omega^{(t)}_{e0,1}$ and proposal variance $s^2_{e0,1}$, that satisfies the above constraints and so hence we effectively are drawing from a truncated Normal distribution. Note that, as described in Browne et al. (2002), the value $s^2_{e0,1}$ can be set by the MLwiN adaptive scheme. Then the update step is as follows:

$$\Omega^{(t+1)}_{e0,1} = \Omega^*_{e0,1}, \quad \text{with probability } \min\left[1, R\frac{p(\Omega^*_{e0,1}|y,\beta)}{p(\Omega^{(t)}_{e0,1}|y,\beta)}\right],$$

$$\Omega^{(t+1)}_{e0,1} = \Omega^{(t)}_{e0,1}, \quad \text{otherwise.}$$

$$\text{Here} \quad R = \frac{\Phi\left(\frac{m_u - \Omega^*_{e0,1}}{s^2_{e0,1}}\right) - \Phi\left(\frac{m_l - \Omega^*_{e0,1}}{s^2_{e0,1}}\right)}{\Phi\left(\frac{m_u - \Omega^{(t)}_{e0,1}}{s^2_{e0,1}}\right) - \Phi\left(\frac{m_l - \Omega^{(t)}_{e0,1}}{s^2_{e0,1}}\right)} \quad \text{which is the Hastings ratio.}$$

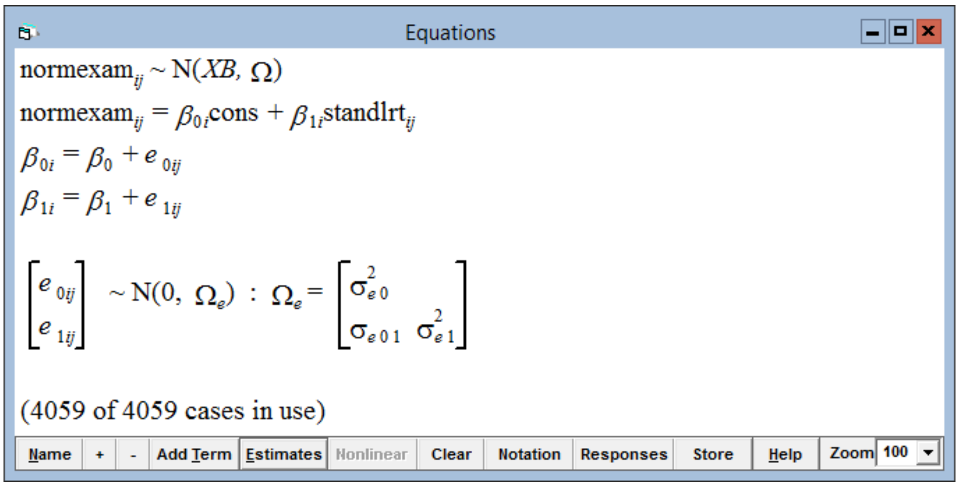
The steps for the other two parameters, $\Omega_{e0,0}$ and $\Omega_{e1,1}$ have similar forms to the above. Now that we have described the two steps the MCMC algorithm as usual consists of repeated application of the steps in turn. We will now explain how to set up this model in MLwiN using the **tutorial** dataset.

9.2 Setting up the model in MLwiN

We will start by setting up the first model from Chapter 2, which has fixed effects defined for **cons** and **standlrt** and **cons** defined as random at level 1. If you are unsure of how to set up this model refer to Chapter 2. Next we need to include the complex variation terms at level 1.

- Select the **standlrt** predictor in the **Equations** window.
- From the **X variable** window that appears click on **(i)student**.
- Click on the **Done** button.

The **Equations** window should now look as follows:


Equations

$$\text{normexam}_{ij} \sim \text{N}(XB, \Omega)$$

$$\text{normexam}_{ij} = \beta_{0i}\text{cons} + \beta_{1i}\text{standlrt}_{ij}$$

$$\beta_{0i} = \beta_0 + e_{0ij}$$

$$\beta_{1i} = \beta_1 + e_{1ij}$$

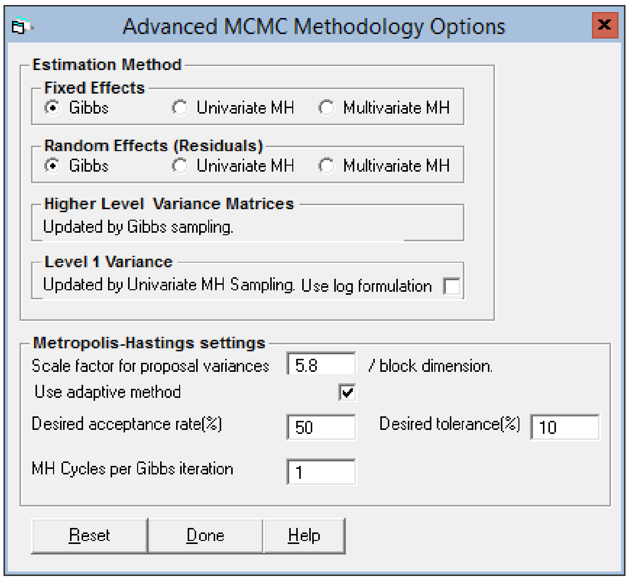
$$\begin{bmatrix} e_{0ij} \\ e_{1ij} \end{bmatrix} \sim \text{N}(0, \Omega_e) : \Omega_e = \begin{bmatrix} \sigma^2_{e0} & \\ \sigma_{e01} & \sigma^2_{e1} \end{bmatrix}$$

(4059 of 4059 cases in use)

We will now need to run the IGLS method and then change to MCMC.

- Click on the **Start** button.
- Click on **Estimation Control** and select the **MCMC** tab.
- Select **MCMC/MCMC Methods** from the **Model** menu.

The **MCMC methods** window will then look as shown below. Note that MLwiN realizes that we need to use MH sampling for the level 1 variance matrix and so changes method automatically.


Advanced MCMC Methodology Options

Estimation Method

Fixed Effects: [✓ Gibbs] Univariate MH Multivariate MH

Random Effects (Residuals): [✓ Gibbs] Univariate MH Multivariate MH

Higher Level Variance Matrices: Updated by Gibbs sampling.

Level 1 Variance: Updated by Univariate MH Sampling. Use log formulation ☐

Metropolis-Hastings settings

Scale factor for proposal variances: 5.8 / block dimension.

Use adaptive method: ✓

Desired acceptance rate(%): 50 Desired tolerance(%): 10

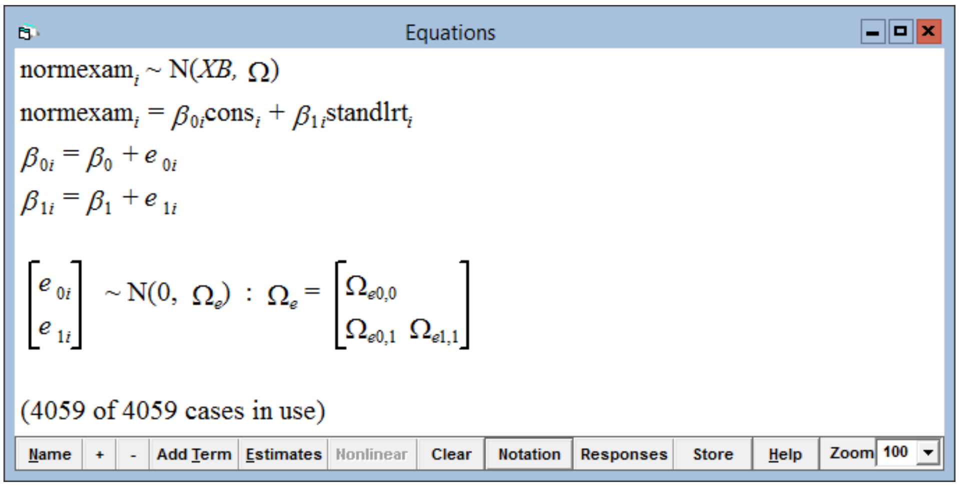
MH Cycles per Gibbs iteration: 1

You may have noticed that the notation we have used earlier in this chapter for the level 1 variance matrix is different to that which you see in the **Equations** window. We can change this:

- Click on the **Notation** button and remove the tick for multiple subscripts.

- Click on the **Done** button.

and we will get the following :



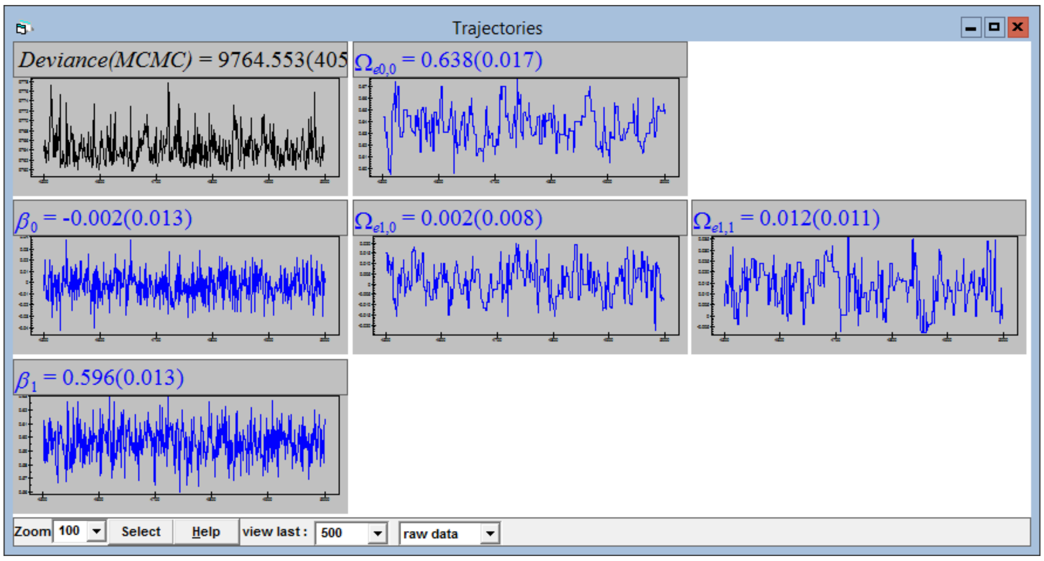
$$\text{normexam}_i \sim \mathrm{N}(XB,\ \Omega)$$
$$\text{normexam}_i = \beta_{0i}\text{cons}_i + \beta_{1i}\text{standlrt}_i$$
$$\beta_{0i} = \beta_0 + e_{0i}$$
$$\beta_{1i} = \beta_1 + e_{1i}$$
$$\begin{bmatrix} e_{0i} \\ e_{1i} \end{bmatrix} \sim \mathrm{N}(0,\ \Omega_e) \ : \ \Omega_e = \begin{bmatrix} \Omega_{e0,0} & \\ \Omega_{e0,1} & \Omega_{e1,1} \end{bmatrix}$$

(4059 of 4059 cases in use)

Now we need to run our model with complex variation.

- Click the **Start** button.

Note that the greater complexity in fitting a complex variance function at level 1 tends to slow computation down and so it will take longer to run this model. Upon running for 5,000 iterations if we open the **Trajectories** window we see the following:



Deviance(MCMC) = 9764.553(405

$\Omega_{e0,0}$ = 0.638(0.017)

β_0 = -0.002(0.013)

$\Omega_{e1,0}$ = 0.002(0.008)

$\Omega_{e1,1}$ = 0.012(0.011)

β_1 = 0.596(0.013)

Here we can tell that the variance parameters are being updated by Metropolis Hastings sampling by the fact that their chains have a more block-like appearance. Also we can see that the chain for parameter $\Omega_{e1,1}$ goes negative so it is more appropriate to use this alternative notation as this term is clearly NOT a variance. We can compare the fit of this model with the simple variance model via the DIC diagnostic:

- Select **MCMC/DIC** diagnostic from the **Model** menu.

The diagnostic is given below and compared with the diagnostic for the model with a constant variance fitted in Chapter 2.

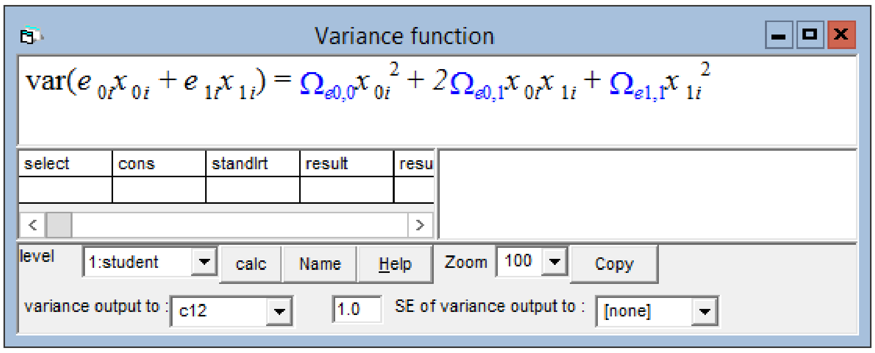
Dbar	D(thetabar)	pD	DIC
9764.55	9759.62	4.93	9769.49
9763.54	9760.51	3.02	9766.56

Here we see that the DIC correctly estimates the 5 parameters in the model and gives a very marginal improvement in fit (9759.6 versus 9760.5). However due to the two additional parameters the DIC value for this model is higher than for the simpler model, suggesting in this case that there is no advantage in fitting the more complex variance function. This is backed up by both the linear and quadratic variance terms being similar to or smaller than their standard errors.

The variance function can still however be calculated by using the **Variance function** window.

- Select **Variance function** from the **Model** menu.
- Select **c12** from the **variance output to** list.
- Press the **calc** button.

Note that the variance function screen should look as follows:



Variance function

$$\mathrm{var}(e_{0i}x_{0i} + e_{1i}x_{1i}) = \Omega_{e0,0}x_{0i}^2 + 2\Omega_{e0,1}x_{0i}x_{1i} + \Omega_{e1,1}x_{1i}^2$$

select	cons	standlrt	result	resu

level: 1:student

Zoom: 100

variance output to: c12

1.0

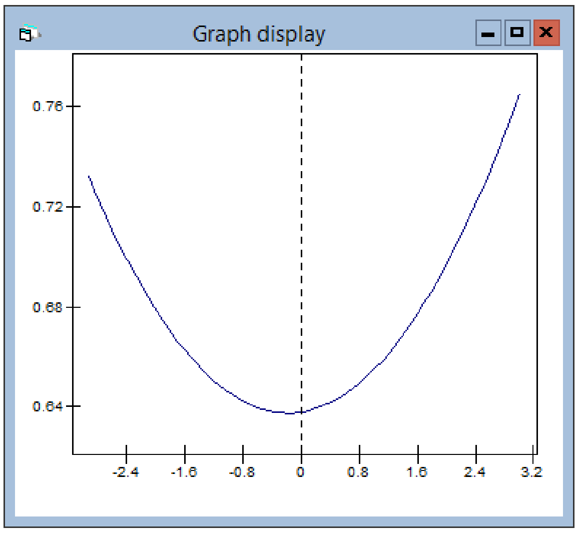
SE of variance output to: [none]

Now we can plot the function we have calculated:

- Select **Customised Graph(s)** from the **Graphs** menu.

- Select column **c12** as the **Y** variable.
- Select column **standlrt** as the **X** variable.
- Select **plot type** line.
- Click on the **Apply** button.

The graph of the function will appear as follows:

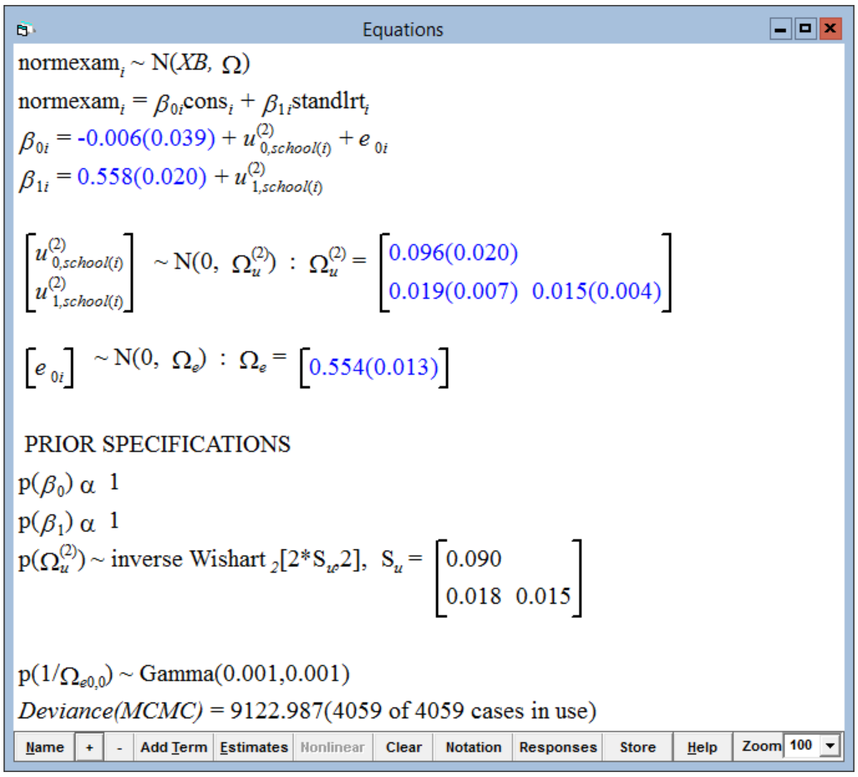


Here we see that this graph mimics the variances of the partitions of the dataset by the categorical intake score that we calculated earlier.

9.3 Complex variance functions in multilevel models

We have in fact already considered a multilevel model that contains complex variation. When we considered the random slopes regression model in Chapter 6 we interpreted this model graphically in terms of non-parallel regression lines for each school. We could however also look at this model in terms of the (total) variance at the school and pupil level depending on the intake score.

If we set up the random slopes model and run it using the default MCMC settings (see Chapter 6 for details on setting up this model) we should get the following:



$\text{normexam}_i \sim \text{N}(XB, \Omega)$

$\text{normexam}_i = \beta_{0i}\text{cons}_i + \beta_{1i}\text{standlrt}_i$

$\beta_{0i} = -0.006(0.039) + u^{(2)}_{0,school(i)} + e_{0i}$

$\beta_{1i} = 0.558(0.020) + u^{(2)}_{1,school(i)}$

$$\begin{bmatrix} u^{(2)}_{0,school(i)} \\ u^{(2)}_{1,school(i)} \end{bmatrix} \sim \text{N}(0, \Omega^{(2)}_u) : \Omega^{(2)}_u = \begin{bmatrix} 0.096(0.020) & \\ 0.019(0.007) & 0.015(0.004) \end{bmatrix}$$

$$\begin{bmatrix} e_{0i} \end{bmatrix} \sim \text{N}(0, \Omega_e) : \Omega_e = \begin{bmatrix} 0.554(0.013) \end{bmatrix}$$

PRIOR SPECIFICATIONS

$p(\beta_0) \propto 1$

$p(\beta_1) \propto 1$

$$p(\Omega^{(2)}_u) \sim \text{inverse Wishart}_2[2*S_u, 2],\ S_u = \begin{bmatrix} 0.090 & \\ 0.018 & 0.015 \end{bmatrix}$$

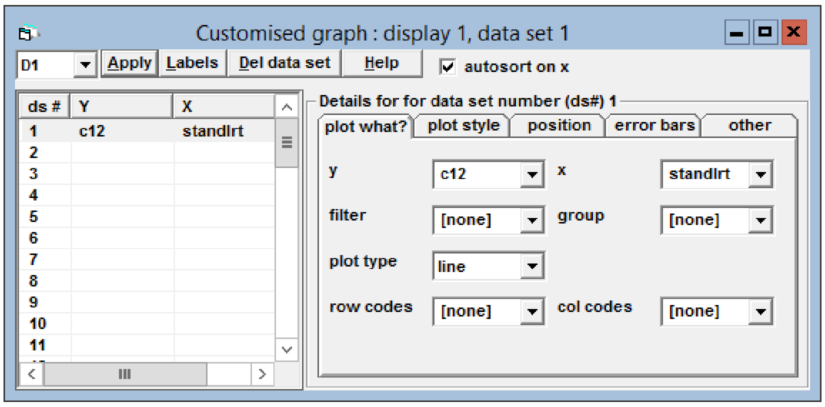
$p(1/\Omega_{e0,0}) \sim \text{Gamma}(0.001, 0.001)$

Deviance(MCMC) = 9122.987(4059 of 4059 cases in use)

Note that as we have clicked on the **Notation** button earlier some of the other terms will have a different notational form. This notation was designed for cross-classified models that will be described later in Chapter 15. Click on **Notation** again and tick the box for multiple subscripts to see standard notation for a 2-level model. Now we can calculate and plot both the level 1 and level 2 variance functions against the intake score by using the **Variance function** window as follows:

- Select **Variance function** from the **Model** menu.
- Select column **c12** from the **variance output to** list.
- Click on the **calc** button.
- Select **2:school** from the **level** list.
- Select column **c13** from the **variance output to** list.
- Click on the **calc** button.

This will set up the level 1 function in column **c12** and the level 2 function in column **c13**. If you have been following this chapter from the start then the level 1 variance function will already be plotted and the **Customised graph** window should be as follows:



Details for for data set number (ds#) 1

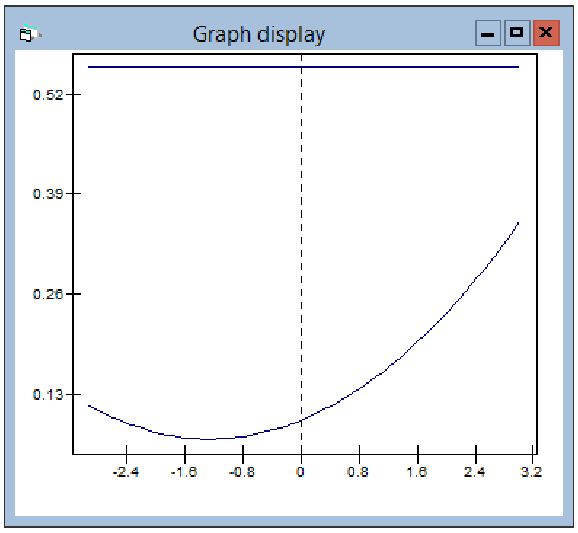
ds #	Y	X
1	c12	standlrt
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		

y: c12 x: standlrt
filter: [none] group: [none]
plot type: line
row codes: [none] col codes: [none]

We now need to specify the level 2 variance function so the following is required:

- Select **ds#2** by clicking in the white box below **Y** next to **2** on the left of the window.
- Select **c13** as the **y** variable.
- Select **standlrt** as the **x** variable.
- Select **plot type** line.
- Click on the **Apply** button.

The graph window will now have two lines in it as shown below:



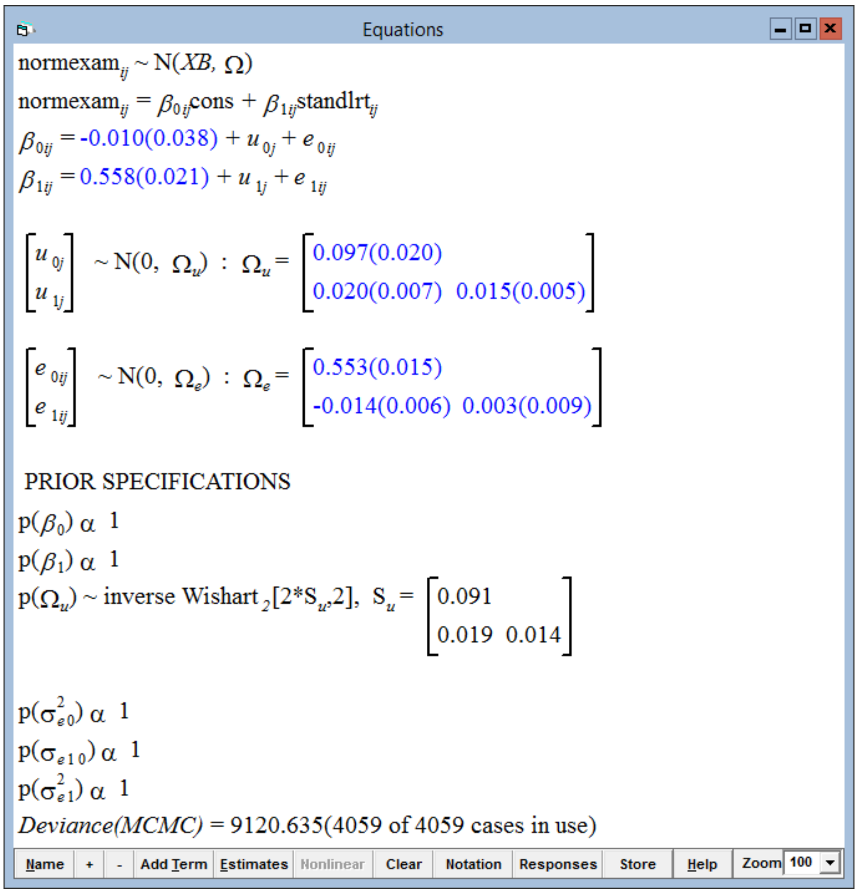
Here the level 1 variance function is the flat (constant) line at 0.554 while the level 2 variance exhibits a quadratic relationship with intake score. Although we did not find any significant evidence of heteroscedasticity in the one level model earlier we can now test if there is any in our two level model. To include the quadratic variance relationship again we need to do the following (after setting estimation method to IGLS):

- Select the **standlrt** predictor in the **Equations** window.
- From the **X variable** window that appears click on **(i)student**.
- Click on the **Done** button.

We can now run this model using MCMC by performing the following:

- Click on the **Start** button.
- Select **MCMC** from the **Estimation** menu then click the **Done** button.
- Click on the **Start** button.

After a short while the model will run and the **Equations** window (after zooming to factor 75 to ensure window all fits on screen) will look as follows:



$\text{normexam}_{ij} \sim N(XB, \Omega)$

$\text{normexam}_{ij} = \beta_{0ij}\text{cons} + \beta_{1ij}\text{standlrt}_{ij}$

$\beta_{0ij} = -0.010(0.038) + u_{0j} + e_{0ij}$

$\beta_{1ij} = 0.558(0.021) + u_{1j} + e_{1ij}$

$$\begin{bmatrix} u_{0j} \\ u_{1j} \end{bmatrix} \sim N(0,\ \Omega_u) \ :\ \Omega_u = \begin{bmatrix} 0.097(0.020) & \\ 0.020(0.007) & 0.015(0.005) \end{bmatrix}$$

$$\begin{bmatrix} e_{0ij} \\ e_{1ij} \end{bmatrix} \sim N(0,\ \Omega_e) \ :\ \Omega_e = \begin{bmatrix} 0.553(0.015) & \\ -0.014(0.006) & 0.003(0.009) \end{bmatrix}$$

PRIOR SPECIFICATIONS

$p(\beta_0) \ \alpha \ 1$

$p(\beta_1) \ \alpha \ 1$

$p(\Omega_u) \sim \text{inverse Wishart}_2[2*S_u, 2],\ S_u = \begin{bmatrix} 0.091 & \\ 0.019 & 0.014 \end{bmatrix}$

$p(\sigma^2_{e0}) \ \alpha \ 1$

$p(\sigma_{e10}) \ \alpha \ 1$

$p(\sigma^2_{e1}) \ \alpha \ 1$

Deviance(MCMC) = 9120.635(4059 of 4059 cases in use)

Here we see that the linear variance coefficient at level 1 (2×-0.014) is negative whilst the quadratic coefficient (0.003) is positive but very small. We can compare the DIC diagnostic for this model with that of the simple random slopes regression model fitted in Chapter 6 by selecting the **MCMC/DIC diagnostic** option on the **Model** window.

Dbar	D(thetabar)	pD	DIC	
9120.64	9028.19	92.44	9213.08	(quadratic variance function at level 1)
9122.99	9031.32	91.67	9214.65	(constant variance at level 1)

Here we see that fitting the model with the complex variance function results in a slight improvement in DIC. The quadratic term here is very small and if the variance function is plotted it looks approximately linear. Therefore it would be interesting to see the effect of explicitly fitting just a linear relationship.

- Change estimation mode to IGLS.
- Click on the quadratic term (0.003) at level 1 in the **Equations** window.
- Reply **Yes** to the question 'Remove term standlrt/standlrt from the level 1 covariance matrix?'.

The blue 0.003 will be replaced by a grey 0 to indicate a structural zero in the equations window. We now need to run this reduced model.

- Click on the **Start** button.
- Select **MCMC** from the **Estimation** menu.
- Click on the **Start** button.
- After the model has finished running select **MCMC/DIC** diagnostic from the **Model** menu.

If we now compare the DIC for this model we see

Dbar	D(thetabar)	pD	DIC	
9119.41	9027.74	91.66	9211.07	(linear variance function at level 1)
9120.64	9028.19	92.44	9213.08	(quadratic variance function at level 1)

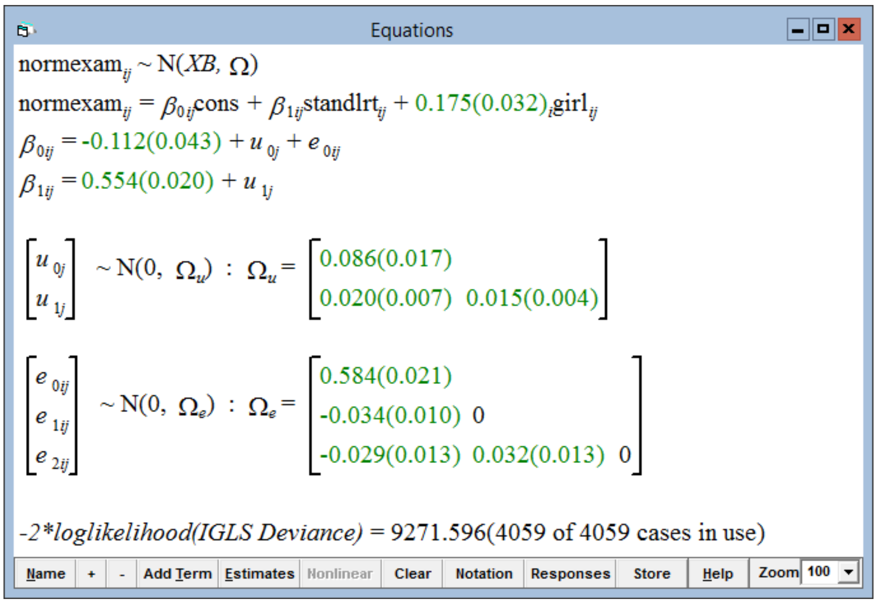
So the simpler linear variance function is prefered to the quadratic function.

9.4 Relationship with gender

At the start of this chapter we considered partitioning the dataset into boys and girls and saw that gender seems to affect both the mean response and the variance of the response. If we continue from the current model which has a linear variance function we can now add gender as a fixed effect and allow the level 1 variance to have terms for gender and gender by intake score. This can be interpreted as fitting two different linear relationships at level 1, one for boys and one for girls. Note that, although we are going to add all the terms at once, generally you would want to add each in turn testing as you go. To set up the model we need to do the following:

- Change **Estimation mode** to IGLS.
- Click on the **Add term** button on the **Equations** window.
- Select **girl** from the **variable** list and click on the **Done** button.
- Click on the **girl** variable in the **Equations** window and tick the level 1 box.
- Click on the bottom right 0.000 in the level 1 variance matrix. A message asking if you want to remove **girl/girl** should appear.
- Choose **Yes** to remove this term.
- Click on the **Start** button.

This will have set up the model and run it using IGLS and the **Equations** window should be as below:



$$\text{normexam}_{ij} \sim \text{N}(XB, \Omega)$$
$$\text{normexam}_{ij} = \beta_{0ij}\text{cons} + \beta_{1ij}\text{standlrt}_{ij} + 0.175(0.032)_i\text{girl}_{ij}$$
$$\beta_{0ij} = -0.112(0.043) + u_{0j} + e_{0ij}$$
$$\beta_{1ij} = 0.554(0.020) + u_{1j}$$
$$\begin{bmatrix} u_{0j} \\ u_{1j} \end{bmatrix} \sim \text{N}(0, \Omega_u) : \Omega_u = \begin{bmatrix} 0.086(0.017) & \\ 0.020(0.007) & 0.015(0.004) \end{bmatrix}$$
$$\begin{bmatrix} e_{0ij} \\ e_{1ij} \\ e_{2ij} \end{bmatrix} \sim \text{N}(0, \Omega_e) : \Omega_e = \begin{bmatrix} 0.584(0.021) & & \\ -0.034(0.010) & 0 & \\ -0.029(0.013) & 0.032(0.013) & 0 \end{bmatrix}$$

*-2*loglikelihood(IGLS Deviance)* = 9271.596(4059 of 4059 cases in use)

We will now run the model using MCMC.

- Change Estimation method to MCMC.
- Click on the **Start** button.

After running the model we can once more check the **DIC diagnostic** value (via the **Model** menu).

Dbar	D(thetabar)	pD	DIC
9083.92	8990.26	93.66	9177.57

Here we see that the DIC diagnostic has reduced significantly. If we had added terms in stages we would have seen that the majority of the reduction was due to adding in gender as a fixed effect but there is also a significant reduction due to the more complex variance structure.

The variance at level 1 is now

$$\sigma^2_{ei} = \Omega_{e0,0} + 2\text{standlrt}_i\Omega_{e0,1} + 2\text{girl}_i\Omega_{e0,2} + 2\text{standlrt}_i\text{girl}_i\Omega_{e1,2}$$

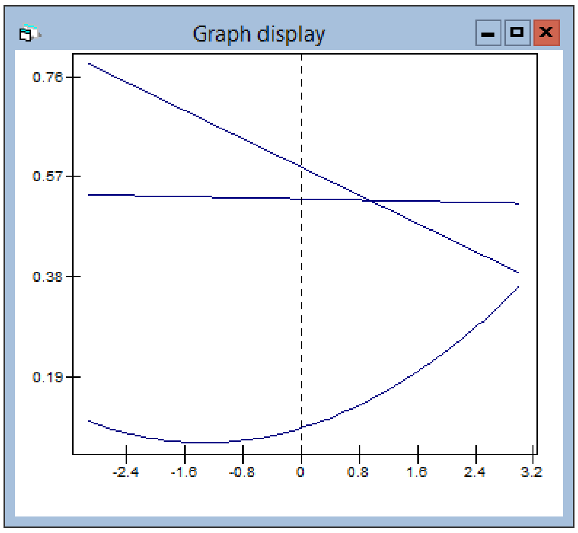
So we have two separate linear relationships for boys and girls, which can be calculated by the Variance function window (along with the updated level 2 variance function)

- Select **Variance function** from the **Model** menu.
- Select **1:student** from the **level** list.
- Select column **c12** from the **variance output to** list.
- Click on the **calc** button.
- Select **2:school** from the **level** list.
- Select column **c13** from the **variance output to** list.
- Click on the **calc** button.

The graph window will still be displaying the correct columns for the variances but as we need to separate out the girls' and boys' lines at level 1 we need to update the **Customized graph** window as follows:

- Select **ds#1** by clicking in the white box below **Y** next to **1** on the left of the window.
- Select **girl** as the **group** column.
- Click on the **Apply** button.

The graph will then appear as follows:



Note that you could label or colour the various lines and rescale the graph to include the origin but this is left as an exercise for the reader. The top two graphs are the level 1 variance functions with the boys having the steeper negative slope. This graph is showing that the choice of school (level 2 variance) is more important for the higher intake pupils than for the lower intake pupils, and that intake score has a much greater effect on the variance of responses for boys than girls. Note that in these complex variance function models the simple intra-school correlation measure described in Chapter 4 does not exist. However for a given gender and intake score we can calculate an equivalent measure, and as in Chapter 4 we can calculate the chain and hence confidence intervals for this measure.

9.5 Alternative log precision formulation

Spiegelhalter et al. (2000*b*) consider the problem of heteroscedasticity in multilevel models in their schools example. One of the problems of the approach we have described in this chapter is the restrictions on prior distributions for fitting this model. Spiegelhalter et al. (2000*b*) get around fitting the variance at level 1 as a function of predictors, by instead fitting the log of the precision at level 1 as a function of predictors. This means that, as the log of the precision can take values anywhere on the real line, there are no constraints to worry about. We can consider fitting the above model using this formulation and so we will have

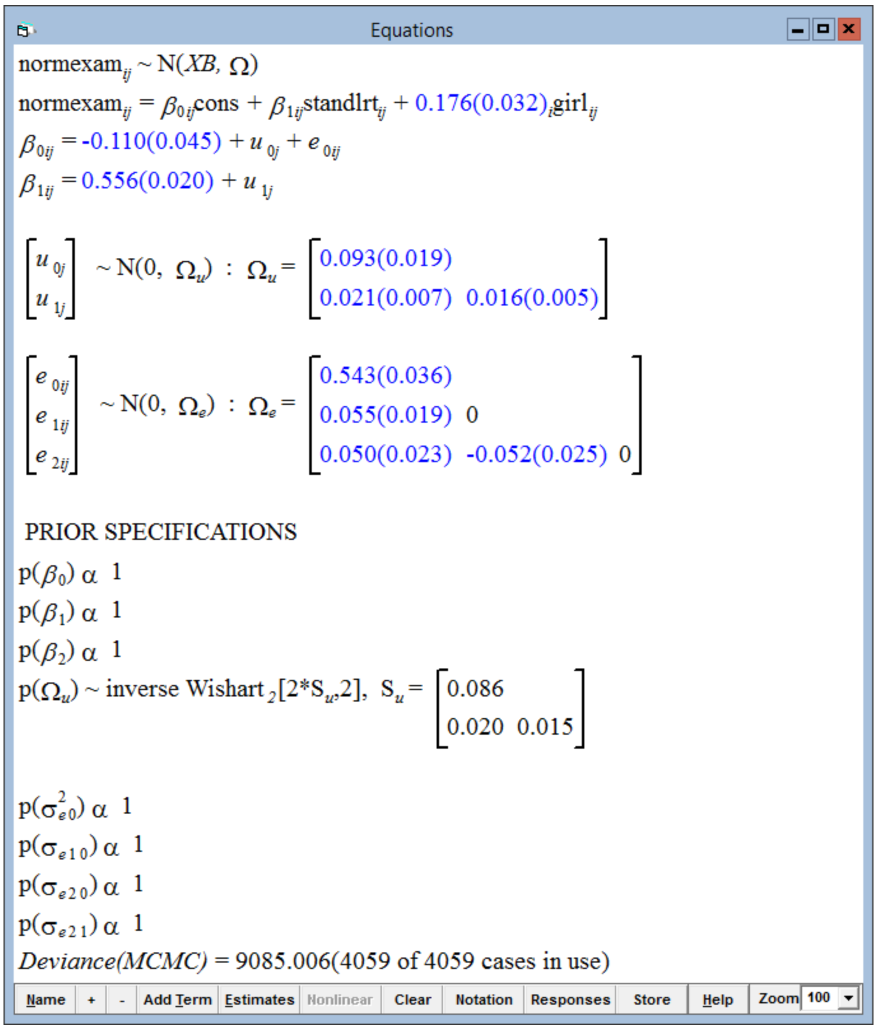
$$\log(1/\sigma^2_{ei}) = \Omega_{e0,0} + 2\text{standlrt}_i\Omega_{e0,1} + 2\text{girl}_i\Omega_{e0,2} + 2\text{standlrt}_i\text{girl}_i\Omega_{e1,2}$$

To fit such a model in MLwiN we need only tell the software that we want to fit this formulation via the **MCMC Methods** window.

- Change estimation to IGLS.
- Click on the **Start** button to run the model.
- Change estimation to MCMC.
- Select **MCMC/MCMC** Methods window from the **Model** menu.
- Click on the **Use log formulation** tickbox under **Level 1 variance**.

It should be noted at this point that currently the **Equations** window does not show this change in the model. For prior distributions MLwiN only allows uniform priors for all elements of Ω_e, although now that the parameter constraints have been removed it is possible to use informative Normal priors via WinBUGS and the WinBUGS interface.

We will now run the model by clicking on the **Start** button. Note that the log formulation is slower to run. Upon completion we will get the following estimates:



$\text{normexam}_{ij} \sim \text{N}(XB, \Omega)$

$\text{normexam}_{ij} = \beta_{0ij}\text{cons} + \beta_{1ij}\text{standlrt}_{ij} + 0.176(0.032)\text{girl}_{ij}$

$\beta_{0ij} = -0.110(0.045) + u_{0j} + e_{0ij}$

$\beta_{1ij} = 0.556(0.020) + u_{1j}$

$$\begin{bmatrix} u_{0j} \\ u_{1j} \end{bmatrix} \sim \text{N}(0,\ \Omega_u) \ :\ \Omega_u = \begin{bmatrix} 0.093(0.019) & \\ 0.021(0.007) & 0.016(0.005) \end{bmatrix}$$

$$\begin{bmatrix} e_{0ij} \\ e_{1ij} \\ e_{2ij} \end{bmatrix} \sim \text{N}(0,\ \Omega_e) \ :\ \Omega_e = \begin{bmatrix} 0.543(0.036) & & \\ 0.055(0.019) & 0 & \\ 0.050(0.023) & -0.052(0.025) & 0 \end{bmatrix}$$

PRIOR SPECIFICATIONS

$p(\beta_0) \propto 1$

$p(\beta_1) \propto 1$

$p(\beta_2) \propto 1$

$$p(\Omega_u) \sim \text{inverse Wishart}_2[2*S_u, 2],\ S_u = \begin{bmatrix} 0.086 & \\ 0.020 & 0.015 \end{bmatrix}$$

$p(\sigma^2_{e0}) \propto 1$

$p(\sigma_{e10}) \propto 1$

$p(\sigma_{e20}) \propto 1$

$p(\sigma_{e21}) \propto 1$

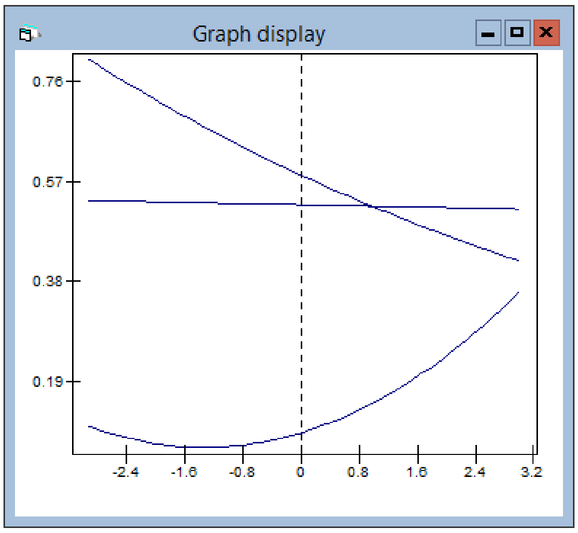
Deviance(MCMC) = 9085.006(4059 of 4059 cases in use)

We can once again construct the graphs for these variance functions via the variance functions window. We will however have to convert these values from inverse precisions to variances.

- Select **Variance functions** from the **Model** menu.
- Select **1:student** from the **level** list
- Select column **c12** from the **variance output to** list.
- Click on the **calc** button.
- Select **2:school** from the **level** list.
- Select column **c13** from the **variance output to** list.
- Click on the **calc** button.
- Select the **Command interface** window.
- Enter the command

```
► CALC C12=1.0/EXPO(C12)
```

The **Graph display** window will now have changed to show the variances in this new model as follows:



Here the relationship at level 1 is still nearly linear. If we look at the DIC diagnostic we see that there is no advantage in using the log-precision formulation and in this case the standard variance function formulation has lower DIC value.

Dbar	D(thetabar)	pD	DIC	
9085.01	8991.09	93.92	9178.93	(log-precision formulation)
9083.92	8990.26	93.66	9177.57	(variance formulation)

Although the log formulation has the advantage of the ability to specify informative priors for level 1 variance terms (note this is available in WinBUGS only) it suffers from the disadvantage that individual coefficients of the variance function do not have a simple interpretation and the real variance relationship is only accessible via a graph as shown above.

Chapter learning outcomes

- ⋆ How to account for heteroscedasticity in Normal response models.
- ⋆ How to graphically interpret variance functions.
- ⋆ How to compare the fit of models via the DIC diagnostic.
- ⋆ How to fit alternative log-precision functions.

Chapter 10

Modelling Binary Responses

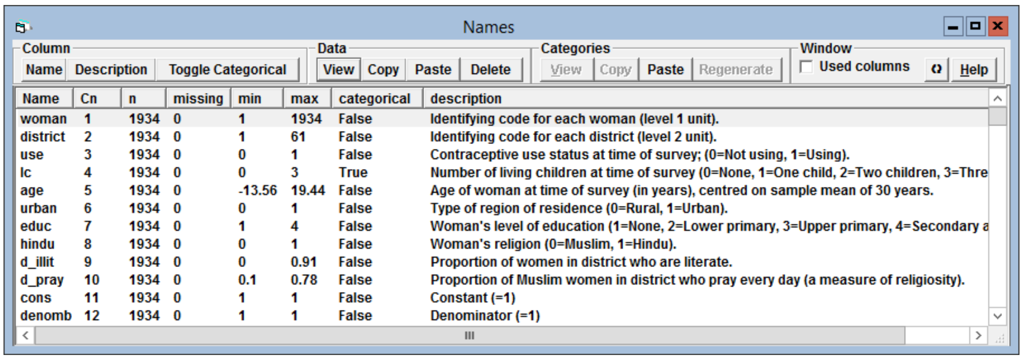
In this book we have so far considered fitting models where our response is a continuous variable. Of course there are many other possible response types, for example interest often lies in binary, proportion or count data. In this chapter we consider binary data. There are many possible scenarios where binary or 0/1 data occur: in education, exam score responses may often be in the form of a pass or fail; in health care, responses are often whether a treatment is successful or not; in political data, whether people vote for a particular party or not.

In this chapter we will consider an example dataset from the 1988 Bangladesh Fertility Survey. This dataset consists of 1934 women who are grouped in 60 districts and the response of interest is whether these women were using contraception at the time of the survey. As predictor variables we will consider effects for the age of the women, the number of children they have and the district they come from. We will also be interested in whether the between-district variation differs for urban and rural areas.

The dataset can be found in the file **bang1.ws**. The reader unfamiliar with the IGLS methods for fitting binary response models is referred to the User's Guide to MLwiN.

- Select **Open Sample Worksheet** from the **File** menu.
- Select **bang1.ws** from the list of worksheets.
- Select **Open**.

The **Names** window will then look as follows:



Name	Cn	n	missing	min	max	categorical	description
woman	1	1934	0	1	1934	False	Identifying code for each woman (level 1 unit).
district	2	1934	0	1	61	False	Identifying code for each district (level 2 unit).
use	3	1934	0	0	1	False	Contraceptive use status at time of survey; (0=Not using, 1=Using).
lc	4	1934	0	0	3	True	Number of living children at time of survey (0=None, 1=One child, 2=Two children, 3=Thre
age	5	1934	0	-13.56	19.44	False	Age of woman at time of survey (in years), centred on sample mean of 30 years.
urban	6	1934	0	0	1	False	Type of region of residence (0=Rural, 1=Urban).
educ	7	1934	0	1	4	False	Woman's level of education (1=None, 2=Lower primary, 3=Upper primary, 4=Secondary a
hindu	8	1934	0	0	1	False	Woman's religion (0=Muslim, 1=Hindu).
d_illit	9	1934	0	0	0.91	False	Proportion of women in district who are literate.
d_pray	10	1934	0	0.1	0.78	False	Proportion of Muslim women in district who pray every day (a measure of religiosity).
cons	11	1934	0	1	1	False	Constant (=1)
denomb	12	1934	0	1	1	False	Denominator (=1)

Here **woman** identifies the individual women, **district** the district they live in and **use** whether they use contraception (1) or not (0). The predictors we will consider are **age** which is the woman's age in years centred around the average age, **lc** which is the number of living children and **urban** which categorises the area (which is a sub-area of the districts) in which they live as either urban (1) or rural (0). The other possible predictors that can be investigated by the reader are **educ** which categorises education level of the women from none (1) to at least secondary education (4), **hindu** which categorises religion into Hindu or Muslim (other religions were excluded), and two district level predictors **d_illit** and **d_pray** which give the proportion of illiterate women and the proportion of women who pray every day, for each district.

10.1 Simple logistic regression model

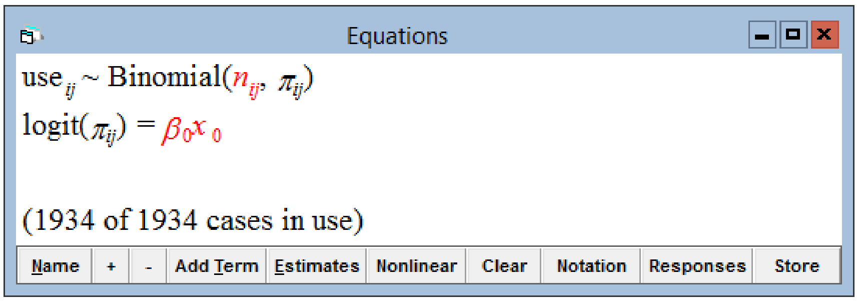
We will start by considering a simple logistic regression model accounting for the age of the women only. Logistic regression models are more complicated to fit than Normal models using the IGLS estimation method in MLwiN as the methods use Taylor series expansions to approximate the model at each iteration. This means that the estimates they produce are quasi-likelihood estimates rather than maximum likelihood. For users familiar with earlier versions of MLwiN the use of two constant columns here labeled **cons** and **bcons** to allow for the transformations performed by the approximation in discrete response modelling is now automated. Binomial response models also need a special (denominator) column that contains the counts of the number of trials each Binomial is based on. For 0/1 data this will be another columns of ones but for proportion data this will contain the number of units on which each proportion is based.

To set up the basic logistic regression model we need to do the following:

- Select **Equations** from the **Model** menu.
- Click on the **Clear** button to remove any existing model in the worksheet

- Click on the red y.
- Select **use** for the **y** variable.
- Select **2-ij** for the **number (N) of levels**.
- Select **district** as **level 2(j)**.
- Select **woman** as **level 1(i)**.
- Click on the **Done** button.
- Click on the **N** and instead choose **Binomial** from the list.

This will set up the response variable and its type, and the hierarchical structure of the dataset. The window will look as follows:



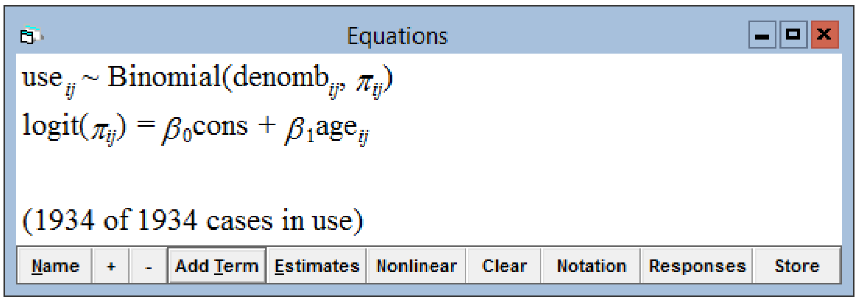
$\text{use}_{ij} \sim \text{Binomial}(n_{ij}, \pi_{ij})$
$\text{logit}(\pi_{ij}) = \beta_0 x_0$

(1934 of 1934 cases in use)

We now need to set up the denominator column and our predictor variables:

- Click on the red n_{ij}.
- Select **denomb** and click on the **Done** button.
- Click on the red x_0.
- Select **cons** and click on the **Done** button.
- Note the level 1 variance is Binomial and so doesn't need adding.
- Click on the **Add Term** button.
- Select **age** from the **variable** list and click on the **Done** button.

The window now looks as follows:



$\text{use}_{ij} \sim \text{Binomial}(\text{denomb}_{ij}, \pi_{ij})$
$\text{logit}(\pi_{ij}) = \beta_0\text{cons} + \beta_1\text{age}_{ij}$

(1934 of 1934 cases in use)

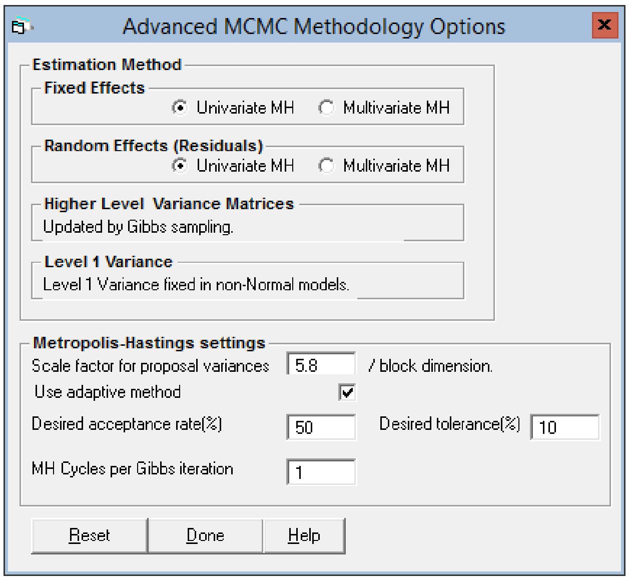
There are several possible quasi-likelihood methods available in MLwiN but, as we are only using them for starting values and to set up the model, we will use the default methods:

- Click on the **Nonlinear** button on the **Equations** window.
- On the window that appears click on **Use Defaults**.
- Click on the **Done** button.
- Click on the **Start** button.

The model should now run in 3 iterations. We now want to run the same model using MCMC.

- Select **MCMC** from the **Estimation** menu.
- Select **MCMC/MCMC Methods** from the **Model** menu.

The window will then appear as shown below. Here we see that for non-Normal responses MLwiN will not allow Gibbs sampling. We will discuss this at the end of the chapter when we compare MLwiN with WinBUGS on binary response models.


Advanced MCMC Methodology Options

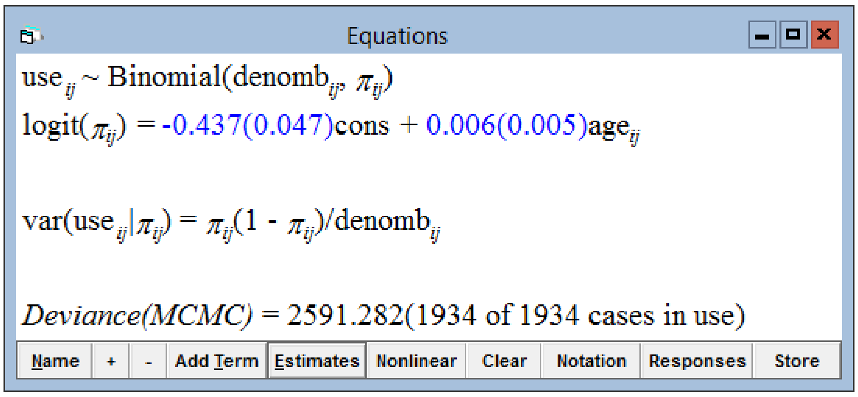
Estimation Method
Fixed Effects: [✓ Univariate MH] Multivariate MH
Random Effects (Residuals): [✓ Univariate MH] Multivariate MH
Higher Level Variance Matrices: Updated by Gibbs sampling.
Level 1 Variance: Level 1 Variance fixed in non-Normal models.

Metropolis-Hastings settings
Scale factor for proposal variances: 5.8 / block dimension.
Use adaptive method: ✓
Desired acceptance rate(%): 50
Desired tolerance(%): 10
MH Cycles per Gibbs iteration: 1

If we now run the model using MCMC:

- Click the **Start** button.

we will get the following results (after clicking twice on the **Estimates** button):



$\text{use}_{ij} \sim \text{Binomial}(\text{denomb}_{ij}, \pi_{ij})$

$\text{logit}(\pi_{ij}) = -0.437(0.047)\text{cons} + 0.006(0.005)\text{age}_{ij}$

$\text{var}(\text{use}_{ij}|\pi_{ij}) = \pi_{ij}(1 - \pi_{ij})/\text{denomb}_{ij}$

Deviance(MCMC) = 2591.282(1934 of 1934 cases in use)

Here we see that the intercept term is -0.437, which, as **age** is centred, corresponds to the average woman. In order to convert this into a probability we need to transform it onto the probability scale. This can be achieved by the anti-logit operation in the **Command interface** window. Typing the command **CALC B1=ALOG (-0.437)** produces a probability of using contraception of 0.392. Note you may have to click the **Output** button to see this in the **Output** window. The age coefficient is small and of the same magnitude as its standard error, suggesting that there is no real effect of age. It should be noted that unlike the quasi-likelihood methods, we can now find a deviance for our models and use the DIC diagnostic.

The deviance formula for a Binomial model is :

$$D = -2\sum_i [y_i \log(p_i) + (1 - y_i)\log(1 - p_i)]$$

where p_i is the predicted value for observation i. To calculate p_i we will need to use the inverse distribution function that corresponds to the link function, so for the logit we will need to calculate the antilogit for each fitted value as described above for the average woman. This is all performed in the background when the DIC diagnostic is calculated.

- Select **MCMC/DIC** diagnostic from the **Model** menu.

The output can be seen below along with that for the simpler model (which we will not fit) with just a constant probability of usage.

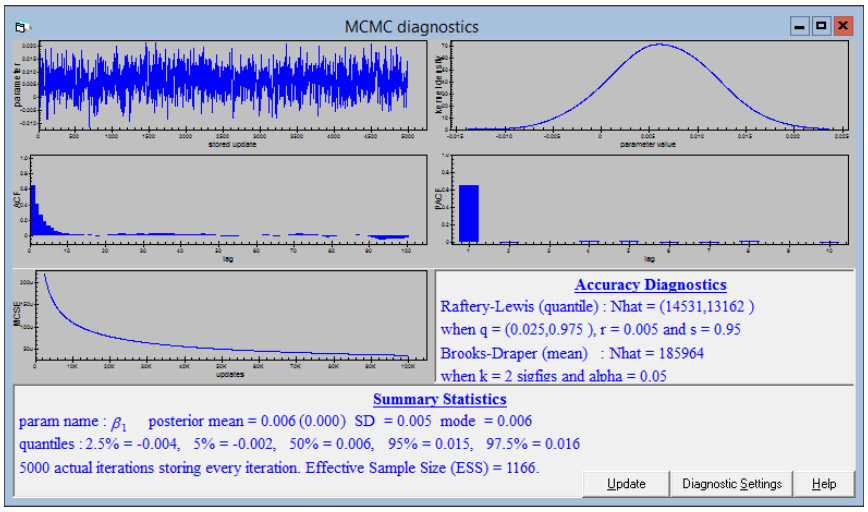
Dbar	D(thetabar)	pD	DIC	
2591.28	2589.29	1.99	2593.27	(Model with age)
2591.90	2590.91	0.99	2592.88	(Model without age)

We can see here that, as with the Normal case, the DIC diagnostic picks up almost exactly the correct degrees of freedom. The DIC diagnostic also confirms that the model without age is only marginally better as we suspected from the coefficients and standard errors. Note that for Binomial models there are two possible parameterisations for D(thetabar), the mean parameterization which uses the average value of p_i from the chain, and the canonical parameterization which uses the average values of the individual β_i that form p_i. In MLwiN we always use the canonical parameterisation but it should be noted that the different parameterizations will often give different DIC values.

In all the models fitted thus far we have been using the default monitoring run length of 5,000 iterations. This is not to be encouraged and we should always check we have run the MCMC algorithm for long enough particularly for non-Normal models where Metropolis sampling is the default for some parameters.

- Select **Trajectories** from the **Model** menu.
- Click on the chain for β_1.
- Select **Yes** to calculate diagnostics.

The diagnostics for the **age** effect will then appear as follows:



Accuracy Diagnostics
Raftery-Lewis (quantile) : Nhat = (14531,13162)
when q = (0.025,0.975), r = 0.005 and s = 0.95
Brooks-Draper (mean) : Nhat = 185964
when k = 2 sigfigs and alpha = 0.05

Summary Statistics
param name : β_1 posterior mean = 0.006 (0.000) SD = 0.005 mode = 0.006
quantiles : 2.5% = -0.004, 5% = -0.002, 50% = 0.006, 95% = 0.015, 97.5% = 0.016
5000 actual iterations storing every iteration. Effective Sample Size (ESS) = 1166.

Here we can see that the kernel plot shows a large probability of a value less than zero. The Raftery-Lewis diagnostic suggests that we should run for roughly three times our current run length. Note that as we are more interested in whether this effect is zero or not than quoting this parameter to two significant figures we will ignore the Brooks-Draper diagnostic.

We can now run this model for 15,000 iterations to see if this changes our estimates:

- Select the **Estimation Control** window.
- Change **Monitoring Chain** length to 15000.
- Click on the **More** button.

If you now look at the DIC diagnostic and the MCMC diagnostics window we see that neither of them have changed much at all. This confirms the results from the shorter run of 5,000 iterations.

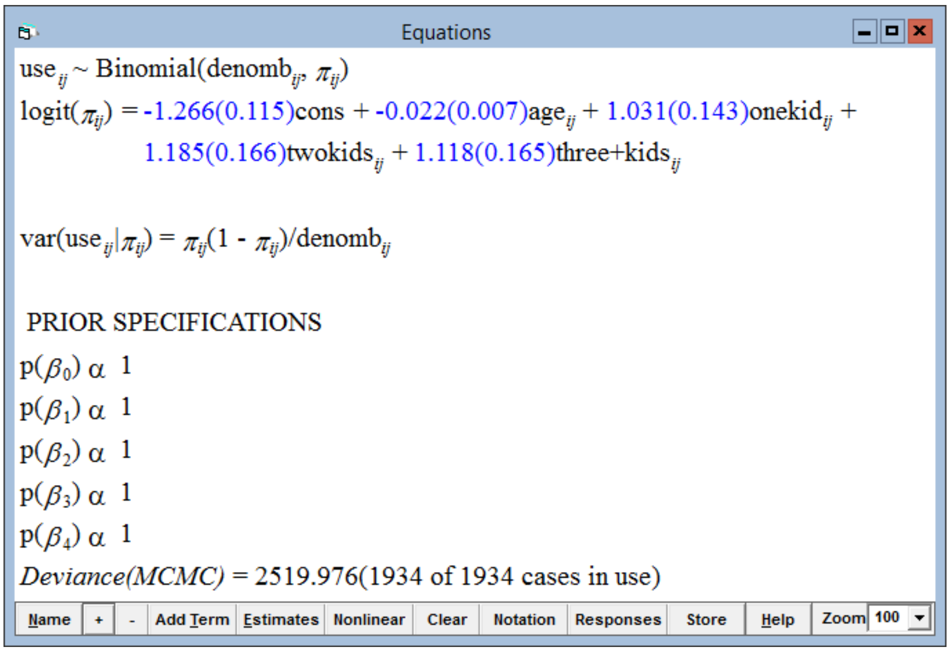
Although we have found that **age** is not significant we will, for now, leave it in the model while adding our next predictor, number of living children (**lc**). We will add this into the model via the **Add Term** button. This can be done (after changing estimation method to IGLS) as follows:

- Change **Estimation mode** to IGLS.
- Click on the **Add Term** button on the **Equations** window.
- Select **lc** from the **variable** pull-down list.
- Click on the **Done** button.

Now we need to run the model using MCMC after using the IGLS method:

- Click on the **Start** button.
- Change **Estimation mode** to MCMC.
- Change **Monitoring Run Length** back to 5000.
- Click on the **Done** button.
- Click on the **Start** button.

After estimation has finished the **Equations** window will (after pressing the + button) look as follows:



$\text{use}_{ij} \sim \text{Binomial}(\text{denomb}_{ij}, \pi_{ij})$

$\text{logit}(\pi_{ij}) = -1.266(0.115)\text{cons} + -0.022(0.007)\text{age}_{ij} + 1.031(0.143)\text{onekid}_{ij} + 1.185(0.166)\text{twokids}_{ij} + 1.118(0.165)\text{three+kids}_{ij}$

$\text{var}(\text{use}_{ij}|\pi_{ij}) = \pi_{ij}(1 - \pi_{ij})/\text{denomb}_{ij}$

PRIOR SPECIFICATIONS

$p(\beta_0) \propto 1$

$p(\beta_1) \propto 1$

$p(\beta_2) \propto 1$

$p(\beta_3) \propto 1$

$p(\beta_4) \propto 1$

Deviance(MCMC) = 2519.976(1934 of 1934 cases in use)

Interestingly when the number of children a woman has is added to the model, the age coefficient now becomes significant, and of opposite sign. This is due to the fact that the number of children is correlated with age.

The DIC diagnostic as shown below is greatly reduced in this model with 5 parameters.

Dbar	D(thetabar)	pD	DIC	
2519.98	2515.10	4.88	2524.85	
2591.28	2589.29	1.99	2593.28	(without number of children)

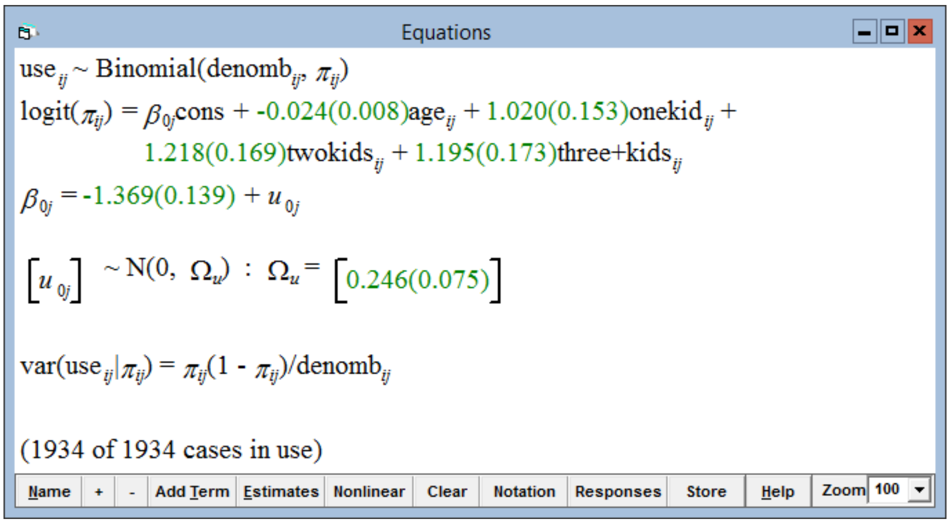
10.2 Random effects logistic regression model

So far we have concentrated on the fixed effects simple logistic regression models analogous to, for the Normal responses, the linear regression models in Chapter 1. As with Normal responses we can also add random effects to our model to account for different probabilities of contraception use for the different districts in which the women live.

To add random effects we use the same procedure as with the Normal models.

- Change **Estimation** method to IGLS.
- Click on **cons** in the **Equations** window.
- Click on the **(j)district** box in the **X variable** window.
- Click on the **Done** button.
- Click on the **Start** button.

Here we have run this model using the IGLS method and the results should look as follows:



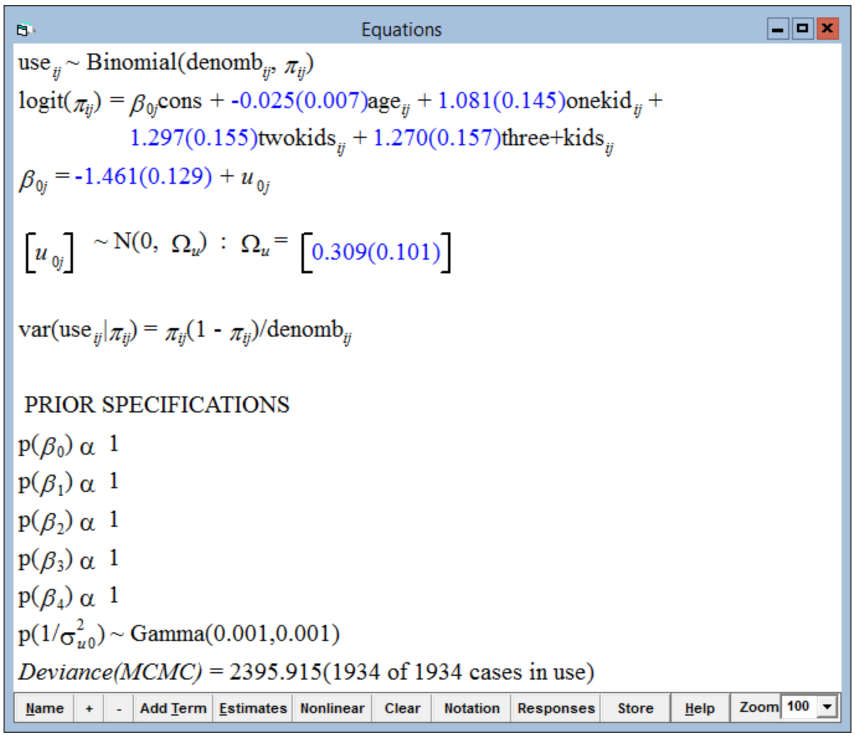
$$\text{use}_{ij} \sim \text{Binomial}(\text{denomb}_{ij}, \pi_{ij})$$
$$\text{logit}(\pi_{ij}) = \beta_{0j}\text{cons} + -0.024(0.008)\text{age}_{ij} + 1.020(0.153)\text{onekid}_{ij} + 1.218(0.169)\text{twokids}_{ij} + 1.195(0.173)\text{three+kids}_{ij}$$
$$\beta_{0j} = -1.369(0.139) + u_{0j}$$
$$\left[u_{0j}\right] \sim \text{N}(0,\ \Omega_u)\ :\ \Omega_u = \left[0.246(0.075)\right]$$
$$\text{var}(\text{use}_{ij}|\pi_{ij}) = \pi_{ij}(1 - \pi_{ij})/\text{denomb}_{ij}$$

(1934 of 1934 cases in use)

To now run this model using MCMC:

- Change **Estimation mode** to MCMC.
- Click on the **Start** button.

After 5,000 iterations we get the following estimates



$\text{use}_{ij} \sim \text{Binomial}(\text{denomb}_{ij}, \pi_{ij})$

$\text{logit}(\pi_{ij}) = \beta_{0j}\text{cons} + -0.025(0.007)\text{age}_{ij} + 1.081(0.145)\text{onekid}_{ij} +$
$1.297(0.155)\text{twokids}_{ij} + 1.270(0.157)\text{three+kids}_{ij}$

$\beta_{0j} = -1.461(0.129) + u_{0j}$

$\left[u_{0j}\right] \sim \text{N}(0, \Omega_u) \; : \; \Omega_u = \left[0.309(0.101)\right]$

$\text{var}(\text{use}_{ij}|\pi_{ij}) = \pi_{ij}(1 - \pi_{ij})/\text{denomb}_{ij}$

PRIOR SPECIFICATIONS

$p(\beta_0) \propto 1$

$p(\beta_1) \propto 1$

$p(\beta_2) \propto 1$

$p(\beta_3) \propto 1$

$p(\beta_4) \propto 1$

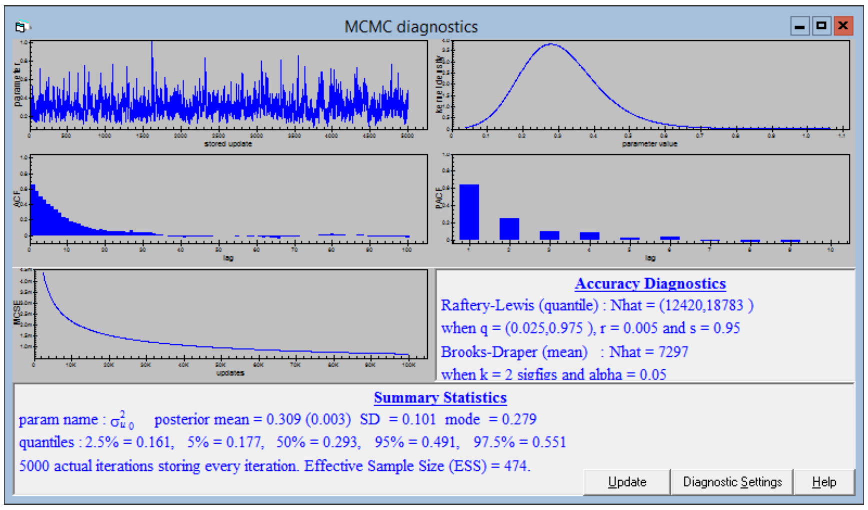
$p(1/\sigma^2_{u0}) \sim \text{Gamma}(0.001,0.001)$

Deviance(MCMC) = 2395.915(1934 of 1934 cases in use)

We can see here that both the fixed effects and the level 2 variance estimate from MCMC are bigger (in magnitude) than the quasi-likelihood estimates. The 1st order MQL method, which is the default method that we have used, is known to give estimates that are biased downwards. We can investigate the level 2 variance in more detail by viewing its MCMC diagnostics.

- Select **Trajectories** from the **Model** menu.
- Click in the trace graph for σ^2_{u0}.
- Select **Yes** to the question 'Calculate MCMC diagnostics?'

This will bring up the MCMC diagnostics for the level 2 variance parameter as shown below. Here we see that the kernel plot has a long right-hand tail as expected and consequently the mode, which is the equivalent to the estimate obtained using quasi-likelihood methods, is smaller than the mean. It has an estimated value 0.279, which compares with the value 0.246 from the 1st order MQL method.



Accuracy Diagnostics
Raftery-Lewis (quantile) : Nhat = (12420,18783)
when q = (0.025,0.975), r = 0.005 and s = 0.95
Brooks-Draper (mean) : Nhat = 7297
when k = 2 sigfigs and alpha = 0.05

Summary Statistics
param name : σ^2_{u0} posterior mean = 0.309 (0.003) SD = 0.101 mode = 0.279
quantiles : 2.5% = 0.161, 5% = 0.177, 50% = 0.293, 95% = 0.491, 97.5% = 0.551
5000 actual iterations storing every iteration. Effective Sample Size (ESS) = 474.

If we look at the DIC diagnostic for the random effects model we get the following:

Dbar	D(thetabar)	pD	DIC	
2395.92	2354.50	41.41	2437.33	(with random effects)
2519.98	2515.10	4.88	2524.85	(without random effects)

So even though we have five fixed effects and sixty random effects due to districts, these sixty district effects are represented by only effectively 36.4 (41.4 − 5) parameters. The DIC diagnostic is reduced by 87.5, so this random effects model is a great improvement, suggesting that there is significantly different contraception usage between the sixty districts.

We can also see from the MCMC diagnostics that the Raftery-Lewis diagnostic suggests that we haven't run the sampler for long enough. Note that if the sampler was run for an extra 15,000 iterations, (which takes a few minutes) this will produce a modal estimate 0.281 which is very similar to the answer after 5,000 iterations.

10.3 Random coefficients for area type

In the dataset, although we have already split up the country of Bangladesh into 60 districts, within these districts there are both urban and rural areas. We have an indicator **urban** that defines whether an individual woman lives in a rural (0) or urban (1) area. We can firstly fit this term as a fixed effect in our model and fit our model using MCMC and 5000 iterations.

- Change **Estimation method** to IGLS.
- Click on the **Add Term** button on the **Equations** window.

- Choose **urban** from the **Variable** list.
- Click on the **Done** button.

Now run the model:

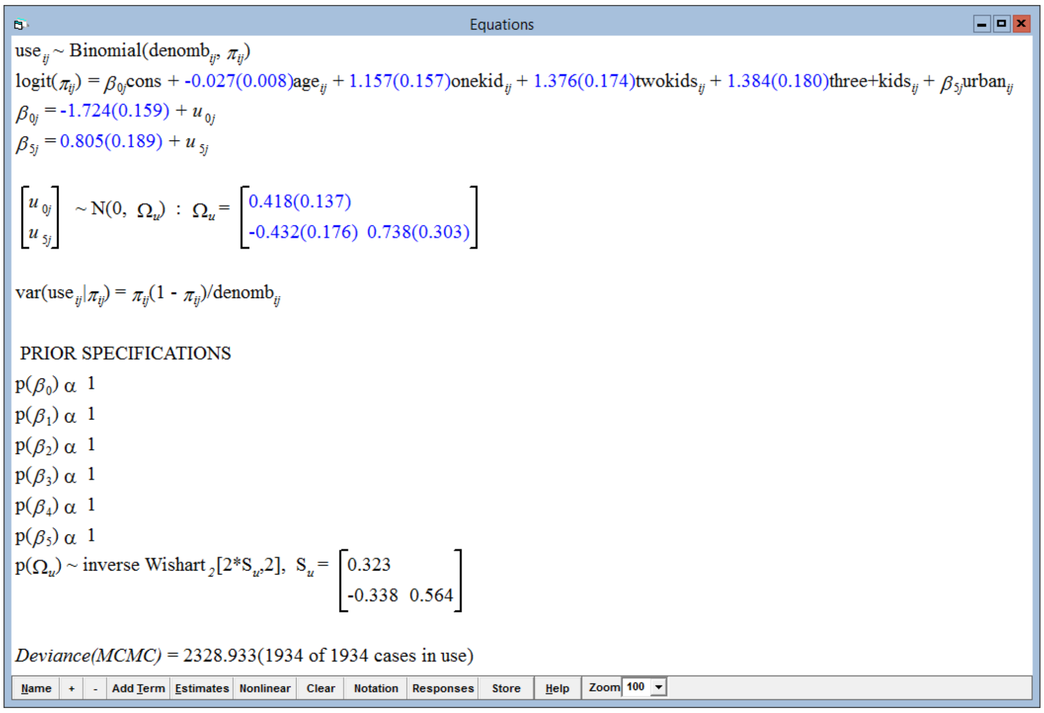
- Click on the **Start** button.
- Change **Estimation Method** to MCMC.
- Click on the **Start** button.

We can now see that **urban** has a fixed coefficient of 0.736 (0.123), which suggests that this is a strong predictor and that women in urban areas are more likely to use contraception than women in rural areas. This is backed up by the reduction in DIC diagnostic to 2408.15 (a drop of 29), which suggests that this is a better model.

Just like fitting random coefficients in a Normal response model, we can now see if the effect of being in an urban area is different for the various districts.

- Change **Estimation method** to IGLS.
- Click on the **urban** predictor.
- From the **X variable** window click in the **j(district)** tickbox.
- Click on the **Done** button.
- Click on the **Start** button.
- Change **Estimation Method** to MCMC.
- Click on the **Start** button.

The model we have now fitted will upon convergence be as follows:


Equations

$\text{use}_{ij} \sim \text{Binomial}(\text{denomb}_{ij}, \pi_{ij})$

$\text{logit}(\pi_{ij}) = \beta_{0j}\text{cons} + -0.027(0.008)\text{age}_{ij} + 1.157(0.157)\text{onekid}_{ij} + 1.376(0.174)\text{twokids}_{ij} + 1.384(0.180)\text{three+kids}_{ij} + \beta_{5j}\text{urban}_{ij}$

$\beta_{0j} = -1.724(0.159) + u_{0j}$

$\beta_{5j} = 0.805(0.189) + u_{5j}$

$$\begin{bmatrix} u_{0j} \\ u_{5j} \end{bmatrix} \sim \text{N}(0,\ \Omega_u)\ :\ \Omega_u = \begin{bmatrix} 0.418(0.137) & \\ -0.432(0.176) & 0.738(0.303) \end{bmatrix}$$

$\text{var}(\text{use}_{ij}|\pi_{ij}) = \pi_{ij}(1 - \pi_{ij})/\text{denomb}_{ij}$

PRIOR SPECIFICATIONS

$p(\beta_0) \propto 1$

$p(\beta_1) \propto 1$

$p(\beta_2) \propto 1$

$p(\beta_3) \propto 1$

$p(\beta_4) \propto 1$

$p(\beta_5) \propto 1$

$$p(\Omega_u) \sim \text{inverse Wishart}_2[2*S_u,2],\ S_u = \begin{bmatrix} 0.323 & \\ -0.338 & 0.564 \end{bmatrix}$$

Deviance(MCMC) = 2328.933(1934 of 1934 cases in use)

Here we see that the variance between districts is different for urban areas and rural areas with the rural areas having a variance of 0.418 and the urban areas having a reduced variance of $0.418 - 2 \times 0.432 + 0.738 = 0.292$. In terms of model fit we are now fitting 6 fixed effects and 2×60 random effects and so if we look at the DIC diagnostic for this model we get:

Dbar	D(thetabar)	pD	DIC	
2328.93	2272.31	56.62	2385.56	(with urban random effects)
2369.36	2330.58	38.79	2408.15	(without urban random effects)

Here once again we see that adding the extra terms increases the effective number of parameters but reduces the DIC diagnostic suggesting that this is a better model.

10.4 Probit regression

The logit link function is only one of the possible link functions that we can use with Binomial data. It is commonly used in medical applications as it has a log-odds interpretation. By this we mean that the exponential of any coefficient of a variable with a fixed effect (x) may be interpreted as an odds ratio, representing the multiplicative effect of a one unit increase in x on the odds of the outcome (if x is continuous) or the odds relative to those for the reference category (if x is a dummy variable). For example the odds of using contraception for a woman with one child (using our last model) are $e^{1.157} = 3.18$ times the odds of using contraception for women with no children assuming all other factors are constant. This is particularly used in medical applications where the response is mortality or infection, and so we can then work out the relative odds of death or infection for two different subsets of the data.

Another popular link function is the probit link, which is the inverse cumulative density function of the Normal distribution. This link is often used for economics applications although, as with the logit, it can be used in many application areas. One important advantage of the probit both in terms of MCMC algorithms and interpretation is that we can think of our response as a threshold from an underlying (unknown) continuous response. This interpretation of a binary response can be used with any link function, but when used with the probit link the unknown continuous response is then Normally distributed.

To illustrate this threshold idea, consider an exam that is marked out of 100. (Note that marks out of 100 are not continuous data but are often treated as Normally distributed and are a better approximation to continuity than a pass/fail response.) Then a pass mark may be set at 50 and our response will be whether an individual student passes (gets 50 or above) or fails (gets below 50). So our observed response is the pass/fail indicator but this is really a surrogate for the more informative (unknown) response, which is the actual mark. The hope is that predictors related to the mark out of 100 will also have a similar relationship to the pass/fail response. Of course in certain situations, for example mortality, it is difficult to think of a continuous predictor which is underlying the 0/1 response (it is hard to rate people as being more dead than each other) but we can still use the threshold idea.

The threshold idea goes back a long way in the statistics literature but has been used recently in conjunction with the Gibbs Sampler by Albert & Chib (1993) using a data augmentation algorithm (Tanner & Wong, 1987).

The idea then proceeds as follows: Let us assume we have a binary variable y_i collected for several individuals i, that is a thresholded version of an (unknown) continuous Normally distributed variable y_i^*. Now if we knew the value of y_i^* then we could fit the standard Gibbs sampling algorithm for Normal response models. So we add an extra step into the Gibbs sampling algorithm and generate y_i^* at each iteration from its conditional posterior distribution which is a truncated Normal distribution with mean (in the single level case) XB and variance 1. The truncation point is zero and if y_i is 0, y_i^* has to be negative and if y_i is 1, y_i^* has to be positive.

Then with the augmented dataset of y_i^* generated, the other parameters can be updated as in the Normal models discussed in earlier chapters. It should be noted that this model can also be updated using Metropolis sampling as with the logistic regression model but the Gibbs sampling algorithm is faster and produces less correlated chains (as shown later). Albert & Chib (1993) also give an approximate Gibbs algorithm for the logistic regression model that uses a t distribution with 8 degrees of freedom as an approximation to the logistic distribution but this has not been implemented in MLwiN.

10.5 Running a probit regression in MLwiN

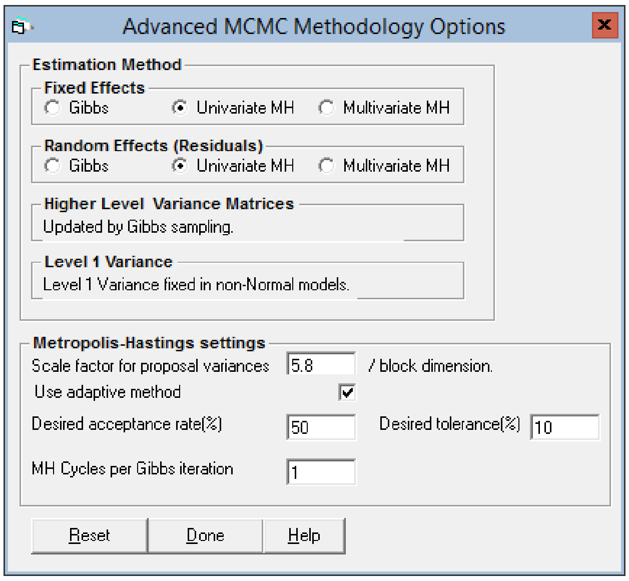
We will now fit the last model with random coefficients for **urban** using a probit link rather than a logit link. To do this we need to do the following:

- Change **Estimation method** to IGLS.
- Click on the word **logit** in the **Equations** window.
- Choose **probit** from the **link function** list and click on the **Done** button.
- Click on the **Start** button.
- Change **Estimation method** to MCMC.

By default MLwiN picks univariate Metropolis Hastings for all non-Normal response models. We will therefore need to change estimation method to Gibbs Sampling.

- Select **MCMC/MCMC Methods** from the **Model** menu.

The window will then look as shown below. Note that MLwiN has realised that it is possible to use Gibbs sampling for this model.


Advanced MCMC Methodology Options

Estimation Method

Fixed Effects: Gibbs [✓ Univariate MH] Multivariate MH

Random Effects (Residuals): Gibbs [✓ Univariate MH] Multivariate MH

Higher Level Variance Matrices: Updated by Gibbs sampling.

Level 1 Variance: Level 1 Variance fixed in non-Normal models.

Metropolis-Hastings settings

Scale factor for proposal variances: 5.8 / block dimension.

Use adaptive method: ✓

Desired acceptance rate(%): 50 Desired tolerance(%): 10

MH Cycles per Gibbs iteration: 1

If we now click on the **Reset** button then MLwiN will choose Gibbs sampling (alternatively simply click on the two Gibbs buttons).

If we now run the model by clicking on the **Start** button, you will notice that we get the message 'Burning In. . . '. This is because we are running Gibbs sampling and there is therefore no need to run an adapting period. The following table gives point estimates and effective sample sizes for runs of 5,000 iterations using both Gibbs and Metropolis sampling for this model.

Parameter	**Gibbs**	**ESS (Gibbs)**	**Metropolis**	**ESS(MH)**
β_0	-1.039 (0.099)	683	-1.048 (0.099)	60
β_1	-0.016 (0.005)	1888	-0.016 (0.005)	275
β_2	0.683 (0.097)	1770	0.692 (0.100)	199
β_3	0.823 (0.106)	1734	0.830 (0.108)	176
β_4	0.819 (0.111)	1584	0.829 (0.108)	93
β_5	0.501 (0.109)	580	0.505 (0.109)	96
Ω_{u00}	0.155 (0.050)	769	0.159 (0.051)	182
Ω_{u05}	-0.160 (0.067)	518	-0.171 (0.066)	141
Ω_{u55}	0.267 (0.118)	414	0.281 (0.111)	124
Time	52s		107s	

From the table we can see that the Gibbs sampler is not only faster but also produces larger effective samples due to less correlation in the chains it produces. We have however roughly the same estimates for both methods.

If we look at the DIC diagnostic for the probit model we see the following:

Dbar	D(thetabar)	pD	DIC	
2328.19	2271.86	56.33	2384.53	(probit)
2328.93	2272.31	56.62	2385.56	(logit)

Here there is very little to choose between the two link functions. It is also possible to use a third link function, the complementary log-log link, with Bernoulli models but this will not be considered here. Note that the Gibbs sampler cannot be used with the probit link when the response is a proportion.

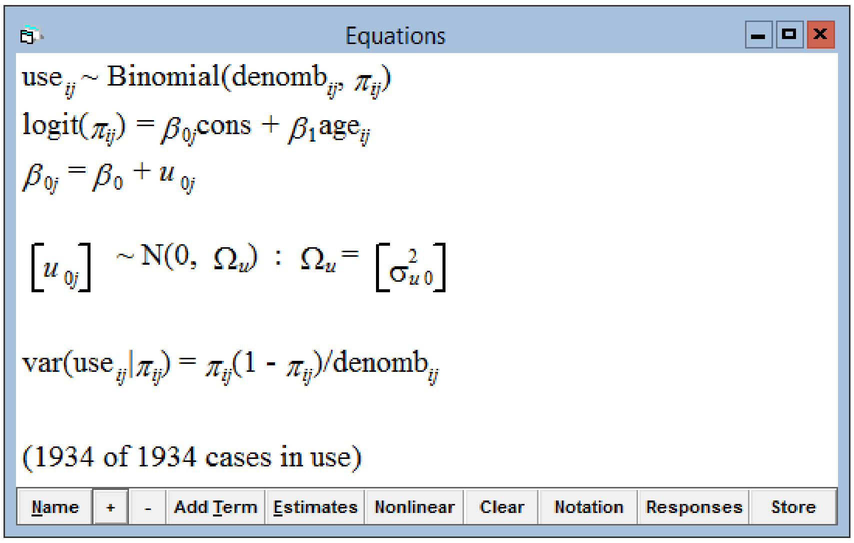
10.6 Comparison with WinBUGS

Binomial response models can be fitted by other MCMC samplers, for example In earlier versions WinBUGS used the Adaptive Rejection (AR) sampling algorithm (Gilks & Wild, 1992). It is often interesting to compare the performance of the various samplers, both in terms of their speed and the autocorrelation in the chains they produce. In the above probit regression example we saw that the data augmentation Gibbs sampler approach was better both in terms of run length and chain correlation than the Metropolis algorithm. This is not however the case when we compare the Metropolis algorithm with the AR algorithm for logistic regression models, as the Metropolis algorithm is usually quicker and so we have to balance greater speed against higher autocorrelation.

We will illustrate this on a simple random slopes logistic regression model. We will clear the current probit regression and set up a model with just an intercept, age as a fixed effect and random effects for the districts.

- Change **Estimation method** to IGLS
- Click on the **Clear** button.
- Click on the red y.
- Select **use** for the **y** variable.
- Select **2-ij** for the **number of levels**.
- Select **district** as **level 2(j)**.
- Select **woman** as **level 1(i)**.
- Click on **Done**.
- Click on the **N** and instead choose **Binomial** from the list.
- Select **logit** from the **link function** list and click on the **Done** button.
- Click on the red n_{ij}.
- Select **denomb** and click on the **Done** button.
- Click on the red x_0.
- Select **cons** and select the **j(district)** tick box and click on the **Done** button.
- Click on the **Add Term** button.
- Select **age** from the variable list and click on the **Done** button.

The model when set up will look as follows:



$\text{use}_{ij} \sim \text{Binomial}(\text{denomb}_{ij}, \pi_{ij})$

$\text{logit}(\pi_{ij}) = \beta_{0j}\text{cons} + \beta_1\text{age}_{ij}$

$\beta_{0j} = \beta_0 + u_{0j}$

$\begin{bmatrix} u_{0j} \end{bmatrix} \sim \text{N}(0,\ \Omega_u) \ : \ \Omega_u = \begin{bmatrix} \sigma^2_{u0} \end{bmatrix}$

$\text{var}(\text{use}_{ij}|\pi_{ij}) = \pi_{ij}(1 - \pi_{ij})/\text{denomb}_{ij}$

(1934 of 1934 cases in use)

Name + - Add Term Estimates Nonlinear Clear Notation Responses Store

We will firstly run IGLS and set up the model for WinBUGS.

- Click on the **Start** button
- Select **MCMC** from the **Estimation** menu.
- Select **MCMC/WinBugs Options** from the **Model** menu.
- Select **WinBugs 1.4** radio button.
- Click on the **Save Current Model in Bugs format** button.
- Change **Save as type** to **.bug files (*.bug)**.
- Save the file as **bang.bug**.

We will assume that you have read Chapter 7 and so are familiar with the basic functionality of WinBUGS. We now need to start the WinBUGS program and load the file **bang.bug** as a text file from the directory it has been saved. Note that again we will need to change the **Files of type** box to **All files (*.*)** to see the file **bang.bug**. When the file is loaded the model definition will look as follows:

```
#WINBUGS 1.4 code generated from MLwiN program

#----MODEL Definition----------------

model
{
# Level 1 definition
for(i in 1:N) {
use[i] ~ dbin(p[i],denom[i])
logit(p[i]) <- beta[1] * cons[i]
+ beta[2] * age[i]
+ u2[district[i]] * cons[i]
}
# Higher level definitions
for (j in 1:n2) {
u2[j] ~ dnorm(0,tau.u2)
}
# Priors for fixed effects
for (k in 1:2) { beta[k] ~ dflat() }
# Priors for random terms
tau.u2 ~ dgamma(0.001000,0.001000)
sigma2.u2 <- 1/tau.u2
}
```

Here we see that our response variable **use** is defined as Binomially distributed and is related to the predictor variables via the logit link function. We will need to repeat the set up procedure that we used in Chapter 7 for the Normal response model:

- Select the **Specification** window from the **Model** menu.
- Click on the **Check Model** button.
- Highlight the **list** identifier at the start of the data list.
- Click on the **Load Data** button.
- Click on the **Compile** button.
- Highlight the **list** identifier at the start of the initial values list.
- Click on the **Load Inits** button.

This will have set up our model and we can now pick our parameters to store. Before we do this we will introduce an interesting feature of WinBUGS not mentioned in Chapter 7. If we wish to find out which methods WinBUGS is using to fit the various components of the model we have defined we can use the **Node Info** tool available from the **Info** menu.

If we then type our node name, for example **beta**, into the **node** box on this window we can then hit the **method** button and get the method used in the **log** window, depending on your defaults you may get:

```
beta[1] UpdaterGLM.LogitUpdater
```

This can be translated to mean that both the fixed effects are being updated using a multivariate Metropolis update using the method developed by Gamerman (Gamerman, 1997). As an aside and in case you do not see the above method WinBUGS currently offers an alternative rejection sampling method. In what follows we will use the Gamerman method but to change to single site you would need to do the following (if you do these instructions you will need to switch back to the Gamerman method to get the same estimates later) :

- Select the **Specification** window from the **Model** menu.
- Click on the **Check Model** button.
- Select the **Blocking options** window from the **Options** menu.
- Remove the **Fixed effects** tick in tick box.
- Highlight the **list** identifier at the start of the data list.
- Click on the **Load Data** button.
- Click on the **Compile** button.
- Highlight the **list** identifier at the start of the initial values list.
- Click on the **Load Inits** button.

Upon asking for methods for **beta** you would then see

```
beta[1] UpdaterRejection.Logit
beta[2] UpdaterRejection.Logit
```

Finally in version 1.3 of WinBUGS the default estimation method was different again and you would see:

```
beta[1] UpdaterDFreeARS.StdUpdater
beta[2] UpdaterDFreeARS.StdUpdater
```

This can be translated to mean that both the fixed effects are being updated by the standard Adaptive Rejection sampler (Gilks & Wild, 1992).

If on the other hand we type the node name **tau.u2** into the **node** box and hit the **method** button we will get in either version:

```
tau.u2 UpdaterGamma.Updater
```

This is because the precision parameter, **tau.u2**, is being updated using Gibbs sampling from a Gamma full conditional distribution.

We now need to tell WinBUGS which parameters we wish to store.

- Select the **Samples** window from the **Inference** menu.
- Change the **beg** value to 501.
- Type **beta** in the **node** box.
- Click on the **Set** button.
- Type **sigma2.u2** in the **node** box.
- Click on the **Set** button.

We have now set up the burn-in of 500 iterations to give a similar burn-in to MLwiN. We have also stated the parameters we wish to store. To run the updates we need to use the update window:

- Select the **Update** window from the **Model** menu.
- Change the **updates** value to 5500.
- Click on the **update** button.

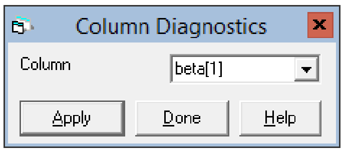
We now need to wait a few minutes for WinBUGS to run. On a Pentium 1.8GHz machine the updates take 79 seconds. In order to measure the efficiency of the sampler we will import the chains back into MLwiN and look at the effective sample size (ESS) measure. To do this we need to do the following:

- Select the **Samples** window from the **Inference** menu.
- Select **beta** from the **node** pull-down list.
- Click on the **Coda** button.
- Click on the window labelled **Coda Index**.
- Choose **Save As** from the **File** menu and choose **plain text (*.txt)** format.
- Save file as **beta.ind**.
- Click on the window labelled **Coda for chain 1**.
- Choose **Save As** from the **File** menu and choose **plain text (*.txt)** format.
- Save file as **beta.out**.

Having saved the two files for the fixed effects we can return to MLwiN and use the **BUGS Options** window (available from the **Model** menu) we used earlier to input the data.

- Change **Input .out file** to **beta.out.txt**
- Change **Input .ind file** to **beta.ind.txt**
- Click on **Input Data** button.

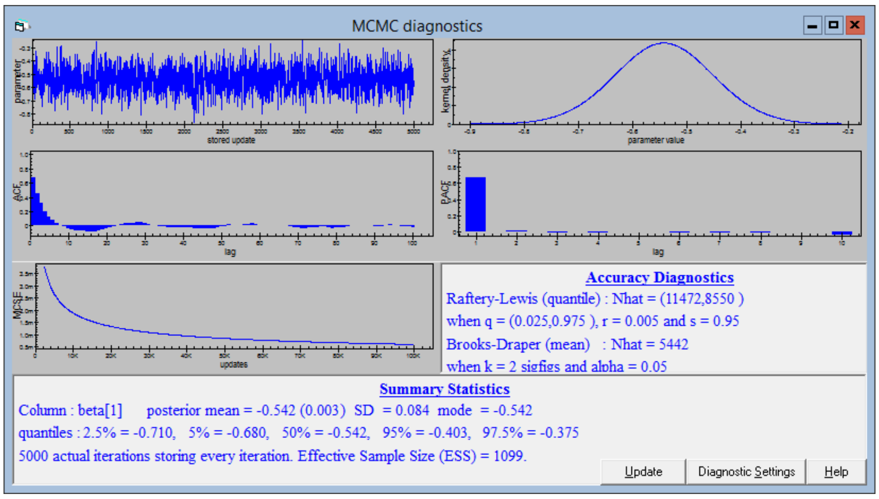
This will store **beta**[**1**] in column **c300** and **beta**[**2**] in column **c301** and rename the columns accordingly. We can now look at the diagnostics for the two parameters via the **Column diagnostics** window:



Column beta[1]

- Select the **Column diagnostics** window from the **Basic Statistics** menu.
- Select **beta**[1] from the **Column** pull-down list.
- Click on the **Apply** button.

The diagnostics will then appear as shown below. Here we see that the effective sample size for β_1 is 1099 for this sample of size 5,000 due to the autocorrelation in the chain. We can repeat this procedure for β_2 and also we can save the chains for σ_u^2 and find its effective sample size. The information for all these parameters will be summarized in the table at the end of this section. Note here that the subscript numbering starts from 1 whilst in MLwiN it starts from zero.



Accuracy Diagnostics
Raftery-Lewis (quantile) : Nhat = (11472,8550)
when q = (0.025,0.975), r = 0.005 and s = 0.95
Brooks-Draper (mean) : Nhat = 5442
when k = 2 sigfigs and alpha = 0.05

Summary Statistics
Column : beta[1] posterior mean = -0.542 (0.003) SD = 0.084 mode = -0.542
quantiles : 2.5% = -0.710, 5% = -0.680, 50% = -0.542, 95% = -0.403, 97.5% = -0.375
5000 actual iterations storing every iteration. Effective Sample Size (ESS) = 1099.

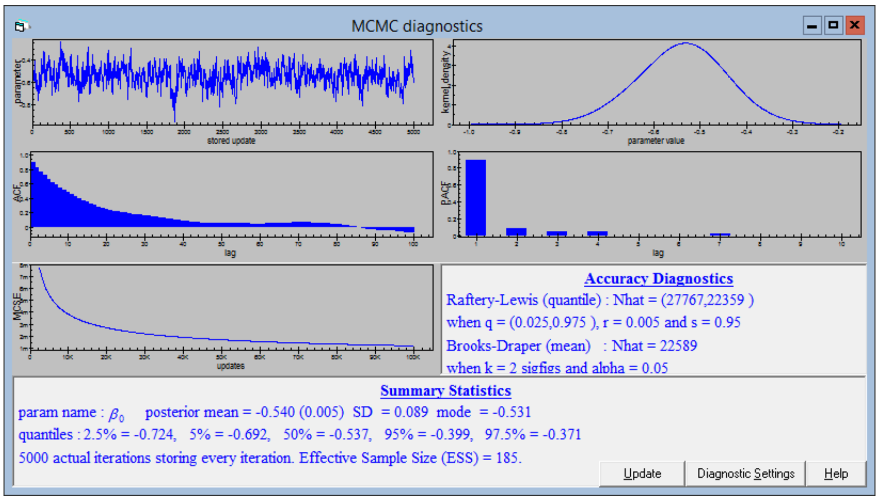
We can now run the same model using Metropolis sampling in MLwiN. The model should currently be set up and MCMC should already be selected and so all we need to do is start the estimation. MLwiN does not by default give a timing estimate so you will need to time estimation via a stopwatch. It is best to close all windows to optimise the speed of execution

- Select **Close All Windows** from the **Windows** menu.
- Click on the **Start** button.

For this example the estimation took 12 seconds on a 1.8GHz Pentium. We can now get effective sample sizes and other information from the MCMC diagnostics window.

- Select **Trajectories** from the **Model** Menu.
- Click in the β_0 trace graph box.
- Select **Yes** in the **Calculate MCMC Diagnostics?** box.

The diagnostics for β_0 using Metropolis sampling will then be displayed as shown below. Here we have higher autocorrelation and so we see that the effective sample size is only 185.



Accuracy Diagnostics

Raftery-Lewis (quantile) : Nhat = (27767,22359)
when q = (0.025,0.975), r = 0.005 and s = 0.95
Brooks-Draper (mean) : Nhat = 22589
when k = 2 sigfigs and alpha = 0.05

Summary Statistics

param name : β_0 posterior mean = -0.540 (0.005) SD = 0.089 mode = -0.531
quantiles : 2.5% = -0.724, 5% = -0.692, 50% = -0.537, 95% = -0.399, 97.5% = -0.371
5000 actual iterations storing every iteration. Effective Sample Size (ESS) = 185.

The results from the two methods can be seen in the following table. The worst mixing parameter is the intercept (β_0) and to get an effective sample size of 5000 will take $8 + (5000/1099)\times$ 71 seconds = 5 minutes 31 seconds using the Gamerman algorithm while Metropolis will take $3 + (5000/185) \times 9$ = 4 minutes 6 seconds. This means that even though the Gamerman algorithm produces a less correlated chain, Metropolis is sufficiently quicker to give an effective sample of 5,000 in less time.

Parameter	**Gamerman**	**ESS**	**Metropolis**	**ESS(MH)**
β_0	-0.542 (0.084)	1099	-0.540 (0.089)	185
β_1	0.009 (0.005)	4929	0.009 (0.005)	1209
σ^2_u	0.265 (0.084)	1221	0.273 (0.092)	433
Time				
Adaptation and burn-in	8s		3s	
Chain	71s		9s	
Total	79s		12s	

So we see that here it appears that Metropolis sampling in MLwiN is better for this random effects logistic regression model. Browne & Draper (2000) showed similar results for a couple of random effects logistic regression models. It is however not guaranteed that this will be true for all models. The Poisson models discussed in the next chapter seem to give highly correlated chains using Metropolis or rejection sampling and they may be models that it makes more sense to fit using the Gamerman algorithm in WinBUGS.

You may want to test the other models fitted in this chapter using WinBUGS, and so we have one word of warning regarding variable names. MLwiN allows virtually any string to represent a column and hence a variable name, for example we have a variable in this chapter called **three+kids**. WinBUGS however would interpret this literally as two variables called **three** and **kids** and would therefore give an error as these variables are not defined in the data section of the code. The WinBUGS interface code therefore strips out such characters so it is worth checking exactly what the variables have been called in your code. It is also therefore sensible to avoid variables that include any of the following symbols: +,-,*,/,ˆ,(,),[and].

Chapter learning outcomes

- ⋆ How to fit models to binary responses.
- ⋆ How to fit multilevel random effects linear models.
- ⋆ How to fit multiple random effects in a logistic model.
- ⋆ How to fit probit regression models using the Gibbs sampler.
- ⋆ How to fit binary response models in WinBUGS.
- ⋆ How to compare estimation methods.
- ⋆ How to time an estimation run in MLwiN.

Chapter 11

Poisson Response Modelling

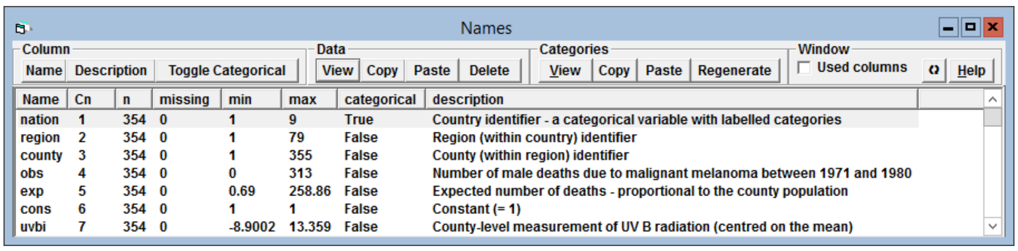
In the last chapter we considered the modelling of binary outcome data. In that case for every individual our response is coded either as a zero or a one, where the meaning of the two states is dependent on the application. For example we may have pass or fail in education applications or presence or absence of a disease in medical applications. Often when large datasets are collected, the response variable (zero or one) may be available at the lowest (individual) level but all other information is collected at a higher level for example at an area level. Then rather than fit a Bernoulli model for the individual responses we may instead either fit a Binomial model for the proportions in each area or a Poisson model for the counts in each area.

In this chapter we will consider fitting models to count data and we will look at a particular example from the public health literature of counts of malignant melanoma mortality in the European community from 1971 to 1980 relating them to exposure to ultra-violet radiation (UVB). This dataset has been investigated more thoroughly in Langford et al. (1998).

The dataset can be found in the MLwiN directory and is called **mmmec.ws**.

- Select **Open Sample Worksheet** from the **File** menu.
- Select **mmmec.ws** and click on **Open**.

The summary of the data will then be displayed as follows:



Name	Cn	n	missing	min	max	categorical	description
nation	1	354	0	1	9	True	Country identifier - a categorical variable with labelled categories
region	2	354	0	1	79	False	Region (within country) identifier
county	3	354	0	1	355	False	County (within region) identifier
obs	4	354	0	0	313	False	Number of male deaths due to malignant melanoma between 1971 and 1980
exp	5	354	0	0.69	258.86	False	Expected number of deaths - proportional to the county population
cons	6	354	0	1	1	False	Constant (= 1)
uvbi	7	354	0	-8.9002	13.359	False	County-level measurement of UV B radiation (centred on the mean)

The data have been collected at a county level and the response variable, **obs**, consists of the observed count of male deaths from malignant melanoma over the ten-year period. There are 354 counties from which the counts have been taken and these counties are taken from 78 regions in 9 countries in the European Community giving us a 3-level structure. We have one predictor also recorded at the county level and this is **uvbi**, which is the (centred) measure of UVB dose that reaches the earth's surface in each county.

Although the data collected are counts of the number of deaths in each county, counties vary in terms of size and number of years in which the data were collected. Therefore if we do not account for these differences in some way any effects we see in our model may actually be picking up the differences in person-years of exposure between the different counties rather than the effects of interest. Commonly therefore in Poisson modelling, the expected counts (**exp** in our worksheet) for each region are calculated. These assume that every individual has the same underlying mortality rate and so the values of **exp** are directly proportional to the person-years of exposure.

As with the Binomial model we need to include a link function to translate the count data to the whole real line. As count data is not restricted to the range 0–1 we do not use the logit link but instead use the log link which will translate the positive counts to values on the whole real line. As we wish to consider (relative) rates rather than counts we need to use a function of the form:

$$\log(\pi_i/\exp_i) = f(\text{predictors}) \qquad \text{where } y_i \sim \text{Poisson}(\pi_i)$$

Here y_i is the observed count, which is assumed to be Poisson distributed with mean parameter π_i. Note that we can rewrite this equation as:

$$\log(\pi_i) = \log(\exp_i) + f(\text{predictors})$$

The first term on the right is an *offset*, a known quantity which is to be included in the equation, and so in MLwiN we can now create the log of the expected counts by doing the following:

- Open the **Command interface** window from the **Data Manipulation** menu and enter the following commands:

```
► calc c9 = loge('exp')
► name c9 'logexp'
```

11.1 Simple Poisson regression model

Now that we have created our offset term we are ready to fit our first Poisson model. We will first fit a single level model that accounts for our main predictor of interest, UV exposure. The model can be set up as follows:

- Select **Equations** from the **Model** Menu.
- Click on the red y.
- Select **obs** as the y variable.
- Select **3-ijk** as the number of levels.
- Select **nation** as **level 3(k)**.
- Select **region** as **level 2(j)**.
- Select **county** as **level 1(i)**.
- Click on the **Done** button.
- Click on the **N** and select **Poisson** from the popup list and click on the **Done** button.

The above commands have set up the response variable, its distribution and the hierarchical structure of the dataset. Note that even though we will firstly fit a single level model we still define the whole three level structure. We now need to set up the offset and predictor variables:

- Click on the π_{ijk} and from the **offset** window that appears select **logexp**.
- Click on the **Done** button.
- Click on the red x_0 and select **cons** and click on the **Done** button.
- Click on the **Add Term** button.
- Select **uvbi** from the **variable** list and click on the **Done** button.
- Click on the **Nonlinear** button.
- Click on the **Use Defaults** button on the **Nonlinear Estimation** window.
- Click on the **Done** button.

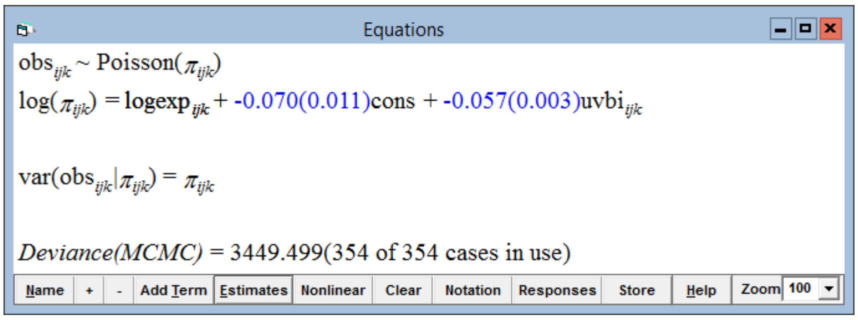
Note that the last three commands set up the default quasi-likelihood method that we will use to get starting values. We can now run the model using MCMC after, as usual, first running IGLS.

- Click on the **Start** button.
- Click on the **Estimation Control** button.

- Click on the **MCMC** tab.
- Change the **Monitoring chain length** to 50,000.
- Change the **refresh rate** to 500.
- Click on the **Done** button.
- Click on the **Start** button.

After running the adapting period the 50,000 iterations should run in a couple of minutes. We have increased the monitoring length here because, as we will see later, the models we fit to this dataset produce parameter chains with high autocorrelations. We will talk briefly at the end of this chapter about other MCMC methods that aim to avoid this problem and we will return to this issue in chapters 23 and 25.

After the 50,000 iterations have completed the **Equations** window should look as follows (Note that you may have to press the **Estimates** and + buttons on the **Equations** window to get exactly this display):



$\text{obs}_{ijk} \sim \text{Poisson}(\pi_{ijk})$

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

$\text{var}(\text{obs}_{ijk}|\pi_{ijk}) = \pi_{ijk}$

Deviance(MCMC) = 3449.499(354 of 354 cases in use)

Rather surprisingly we see that increased UV exposure is associated with a reduction of incidence of melanoma. We will try to explain this by accounting for the hierarchical structure in the data later. The DIC diagnostic is available for Poisson models. Here the deviance takes the following form:

$$D = 2\sum_i (y_i \log \frac{y_i}{e^{\theta_i}} - y_i + e^{\theta_i}), \qquad \text{where } \theta_i = \log(\exp_i) + f(\text{predictors})$$

If we were to look at the DIC diagnostic available from the **MCMC** entry in the **Model** menu we will then see:

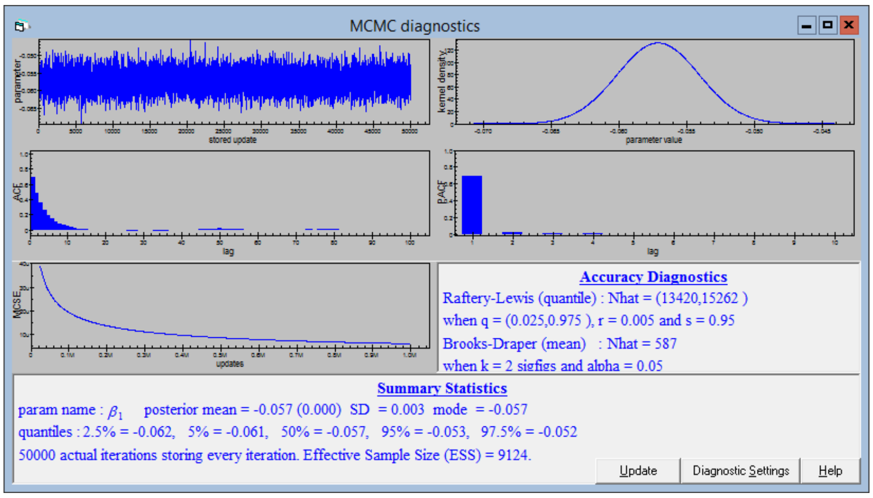
Dbar	D(thetabar)	pD	DIC	
3449.50	3447.55	1.95	3451.45	
3953.36	3952.35	1.01	3954.37	(Without UVBI for comparison)

Here we see that the diagnostic picks up the fact that there are 2 parameters in the model. We also see that adding UV exposure greatly reduces the DIC value suggesting this is a much better model than assuming a common mortality.

We can also look at the diagnostics for the UV effect:

- Select **Trajectories** from the **Model** menu.
- Click on the trace for β_1.
- Click on **Yes** to '**Calculate MCMC diagnostics?**'

The diagnostics will then appear as follows:



Accuracy Diagnostics
Raftery-Lewis (quantile) : Nhat = (13420,15262)
when q = (0.025,0.975), r = 0.005 and s = 0.95
Brooks-Draper (mean) : Nhat = 587
when k = 2 sigfigs and alpha = 0.05

Summary Statistics
param name : β_1 posterior mean = -0.057 (0.000) SD = 0.003 mode = -0.057
quantiles : 2.5% = -0.062, 5% = -0.061, 50% = -0.057, 95% = -0.053, 97.5% = -0.052
50000 actual iterations storing every iteration. Effective Sample Size (ESS) = 9124.

We can see that in this simple model the autocorrelations are fairly small and we did not really need to run for 50,000, although we will for later models. We will now look at including some of the geographical structure in the model.

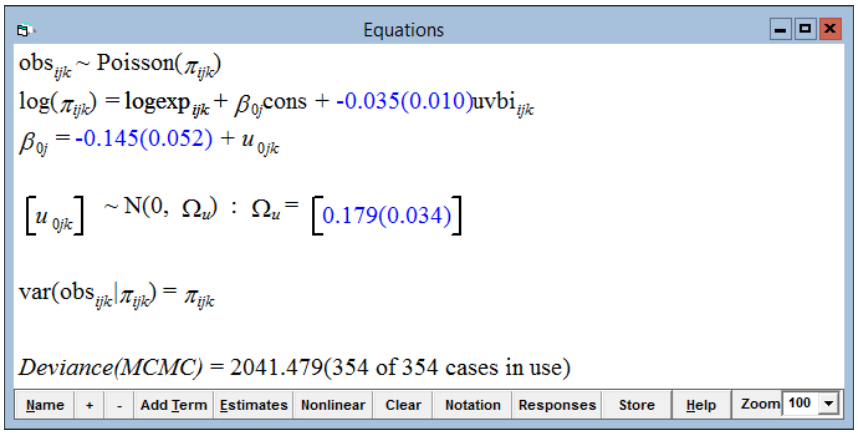
11.2 Adding in region level random effects

The single-level Poisson regression model assumes that the mortality rate is only dependent on the UV exposure of the county and that this relationship is the same for all regions. We can extend a Poisson model to a random effects Poisson model in the same way as for Normal and Binomial response models, by allowing a random effect for each higher level unit, in this case region:

- Change **Estimation method** to **IGLS**.
- In the **Equations** window click on β_0 (**cons**).
- Click in the **(j)region** tick box.
- Click on the **Done** button.
- Click on the **Start** button to obtain the IGLS estimates.
- Change **Estimation method** to **MCMC**.

- Click on the **Start** button.

When the estimation has finished we get the following estimates:



$\text{obs}_{ijk} \sim \text{Poisson}(\pi_{ijk})$

$\log(\pi_{ijk}) = \text{logexp}_{ijk} + \beta_{0j}\text{cons} + -0.035(0.010)\text{uvbi}_{ijk}$

$\beta_{0j} = -0.145(0.052) + u_{0jk}$

$\begin{bmatrix} u_{0jk} \end{bmatrix} \sim \text{N}(0,\ \Omega_u) \ :\ \Omega_u = \begin{bmatrix} 0.179(0.034) \end{bmatrix}$

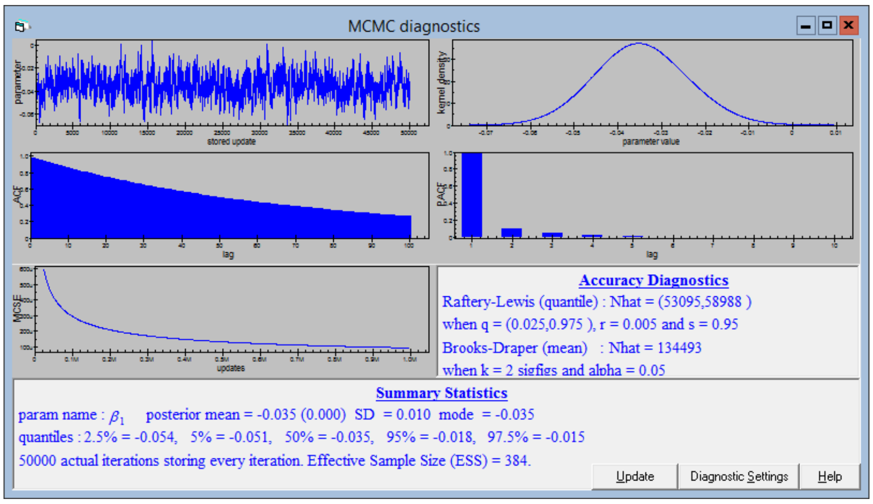
$\text{var}(\text{obs}_{ijk}|\pi_{ijk}) = \pi_{ijk}$

Deviance(MCMC) = 2041.479(354 of 354 cases in use)

Here we see that there is a large variation between the regions and that this variation has reduced the negative effect of UV exposure from −0.057 to −0.034 but that the effect is still significant. If we were to calculate the DIC diagnostic for this model we will get:

Dbar	D(thetabar)	pD	DIC
2041.48	1971.09	70.39	2111.87

This is a huge reduction in DIC showing that this model is a much better fit to the data. The 78 regions are represented by 70.39 effective parameters showing that the region terms are important in the model. Looking again at the diagnostic plots for the **uvbi** predictor we see:



Accuracy Diagnostics
Raftery-Lewis (quantile) : Nhat = (53095,58988)
when q = (0.025,0.975), r = 0.005 and s = 0.95
Brooks-Draper (mean) : Nhat = 134493
when k = 2 sigfigs and alpha = 0.05

Summary Statistics
param name : β_1 posterior mean = -0.035 (0.000) SD = 0.010 mode = -0.035
quantiles : 2.5% = -0.054, 5% = -0.051, 50% = -0.035, 95% = -0.018, 97.5% = -0.015
50000 actual iterations storing every iteration. Effective Sample Size (ESS) = 384.

Here the autocorrelations in the chains are much higher so we this time do need to run for 50,000 iterations. (In fact the Raftery-Lewis diagnostic suggests running for slightly longer.) This is, however, not a problem as 50,000 iterations only takes a couple of minutes to run.

Of course our data structure has an additional level of stratification in that each region is in one of nine countries. We will next consider how to fit these in the model.

11.3 Including nation effects in the model

We will consider two ways of fitting the effects for the nine EU nations into our model. Firstly we will consider fitting nation as random effects in our model. This will mean that our predicted mortality rates will be comprised of a fixed effect for UV exposure and random effects for both the region and nation in which the county lies.

- Change **Estimation method** to **IGLS**.
- In the **Equations** window click on β_0 (**cons**).
- Click in the **(k)nation** tick box.
- Click on the **Done** button.
- Click on the **Start** button to obtain the IGLS estimates.
- Change **Estimation method** to **MCMC**.
- Click on the **Start** button.

After estimation, if we firstly look at the DIC diagnostic for this new model we get:

Dbar	D(thetabar)	pD	DIC	
2039.98	1978.83	61.15	2101.13	
2041.48	1971.09	70.39	2111.87	(without nation effects)

So at first glance our new model has a reduction of DIC diagnostic of about 10 points so is a better model. However if we look at the constituent parts we see that the new model has a worse fit (measured by D(thetabar)) even though it has more parameters! You may be wondering then why the DIC diagnostic thinks that we have a better model. This is because although we have nominally more parameters, the random effects associated with region are less important when the nation effects are taken into account and so the effective number of parameters (pD) drops.

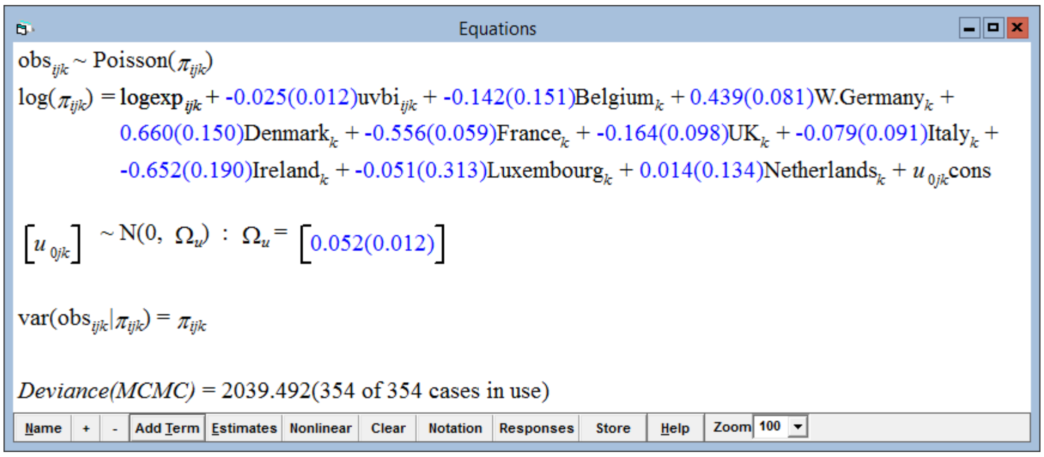
Looking at the **Equations** window we see that the variance between countries is four times the magnitude of the variance between regions but has a large standard error. This may be due to the fact that we only have nine countries in our dataset. It may therefore be more sensible to fit the nation effects as fixed rather than random effects. To do this we will use the **Add term** button on the **Equations** window:

- Change **Estimation method** to **IGLS**.
- In the **Equations** window click on β_0 (**cons**).
- Remove the ticks in the **(k)nation** and **Fixed parameter** tick boxes.
- Click on the **Done** button.
- Click on the **Add Term** button on the **Equations** window.
- Select **nation** from the **variable** dropdown list.
- Select [**none**] from the **reference category** dropdown list.
- Click on the **Done** button.

Generally we add dummy variables and use one category, for example **Belgium**, as a baseline. Here however we have removed the intercept and so can estimate effects for all nine countries. The two model formulations are just reparameterisations of each other and it happens that the parameterisation with no intercept gives less correlated MCMC chains. We now need to fit this model so:

- Click on the **Start** button to obtain the IGLS estimates.
- Change **Estimation method** to **MCMC**.
- Click on the **Start** button.

After 50,000 iterations the **Equations** window will look as follows:



$\text{obs}_{ijk} \sim \text{Poisson}(\pi_{ijk})$

$\log(\pi_{ijk}) = \text{logexp}_{ijk} + -0.025(0.012)\text{uvbi}_{ijk} + -0.142(0.151)\text{Belgium}_k + 0.439(0.081)\text{W.Germany}_k + 0.660(0.150)\text{Denmark}_k + -0.556(0.059)\text{France}_k + -0.164(0.098)\text{UK}_k + -0.079(0.091)\text{Italy}_k + -0.652(0.190)\text{Ireland}_k + -0.051(0.313)\text{Luxembourg}_k + 0.014(0.134)\text{Netherlands}_k + u_{0jk}\text{cons}$

$\left[u_{0jk}\right] \sim \text{N}(0,\ \Omega_u)\ :\ \Omega_u = \left[0.052(0.012)\right]$

$\text{var}(\text{obs}_{ijk}|\pi_{ijk}) = \pi_{ijk}$

Deviance(MCMC) = 2039.492(354 of 354 cases in use)

Here we see that West Germany and Denmark have the highest melanoma mortality rates whilst Ireland and France have the lowest. A comparison of the DIC diagnostic between the fixed and random effects models gives:

Dbar	D(thetabar)	pD	DIC	
2039.49	1977.81	61.68	2101.17	(fixed)
2039.98	1978.83	61.15	2101.13	(random)

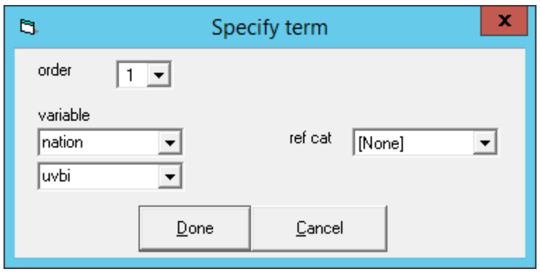
This shows that there is no advantage in treating the nine countries as random effects as opposed to fixed. Note that although we have run for 50,000 iterations some of the parameter chains suggest, through the Raftery-Lewis diagnostic, that we need to run for longer.

11.4 Interaction with UV exposure

We can extend our model further by removing the restriction that the effect of UV exposure is constant across countries after accounting for country and region differences. We can remove the UV term and instead fit separate UV terms for each country. To do this we need to do the following:

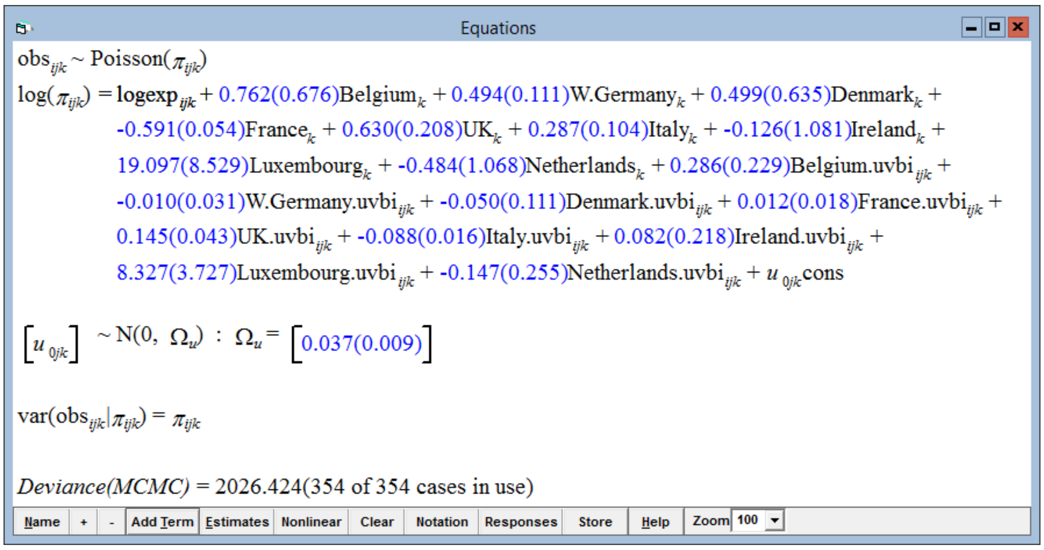
- Change **Estimation method** to IGLS.
- In the **Equations** window click on the β_1 term (**uvbi**).
- Click on the **Delete Term** button.
- Click on the **Add Term** button on the **Equations** window.
- Select **1** in the **order** box.
- Select **nation** from the first **variable** dropdown list.
- Select **uvbi** from the second **variable** dropdown list.
- Select [**none**] from the first **ref cat** dropdown list.

The window should now look as follows:



- Click on the **Done** button.
- Click on the **Start** button.
- Change **Estimation method** to **MCMC**.
- Click on the **Start** button.

After 50,000 iterations we will get the following point estimates:



$\text{obs}_{ijk} \sim \text{Poisson}(\pi_{ijk})$

$\log(\pi_{ijk}) = \text{logexp}_{ijk} + 0.762(0.676)\text{Belgium}_k + 0.494(0.111)\text{W.Germany}_k + 0.499(0.635)\text{Denmark}_k +$
$-0.591(0.054)\text{France}_k + 0.630(0.208)\text{UK}_k + 0.287(0.104)\text{Italy}_k + -0.126(1.081)\text{Ireland}_k +$
$19.097(8.529)\text{Luxembourg}_k + -0.484(1.068)\text{Netherlands}_k + 0.286(0.229)\text{Belgium.uvbi}_{ijk} +$
$-0.010(0.031)\text{W.Germany.uvbi}_{ijk} + -0.050(0.111)\text{Denmark.uvbi}_{ijk} + 0.012(0.018)\text{France.uvbi}_{ijk} +$
$0.145(0.043)\text{UK.uvbi}_{ijk} + -0.088(0.016)\text{Italy.uvbi}_{ijk} + 0.082(0.218)\text{Ireland.uvbi}_{ijk} +$
$8.327(3.727)\text{Luxembourg.uvbi}_{ijk} + -0.147(0.255)\text{Netherlands.uvbi}_{ijk} + u_{0jk}\text{cons}$

$\left[u_{0jk}\right] \sim \text{N}(0,\ \Omega_u) \ : \ \Omega_u = \left[0.037(0.009)\right]$

$\text{var}(\text{obs}_{ijk}|\pi_{ijk}) = \pi_{ijk}$

Deviance(MCMC) = 2026.424(354 of 354 cases in use)

So we now see that France still has a significantly lower mortality rate than average, and Germany a significantly higher mortality rate (at average UVB exposure for the dataset). Both the UK and Italy have significantly higher rates than average (at average UVB exposure for the dataset) when the effect of UV exposure is allowed to vary between countries. The UK has a significant positive effect of UV exposure while Italy has a significant negative effect of UV exposure. There may be many reasons for these findings, for example Italy only has values of UV exposure greater than 2 and so this will imply that for the majority of Italians melanoma mortality is lower than average, and the significant interaction will suggest higher rates of melanoma in the north of Italy than the south. The UK by contrast always has negative values of exposure (remember this variable has been centred) and so the positive coefficient of the exposure in the UK suggests higher rates of melanoma in the south of the UK. One (of many) possible reason for this may be that the south is more affluent and so people there can afford more holidays in sunny places. This is of course speculative and would require matching this dataset with some economic data to back up this hypothesis. It is worth noting here that these interpretations rely on the fact that our offsets are the logs of the expected counts for each region and other forms of offset for example the logs of population size at risk will result in different interpretations.

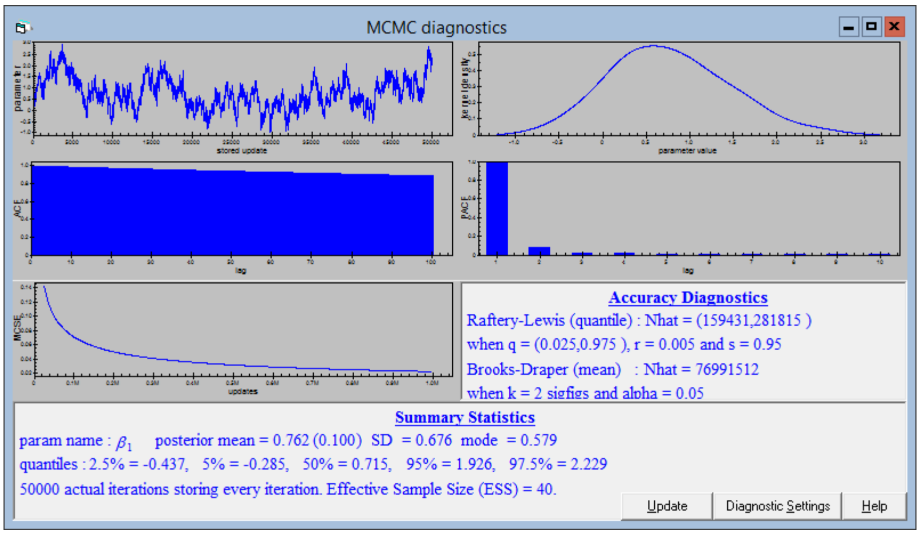
We can compare this model with the last model via the DIC diagnostic.

Dbar	D(thetabar)	pD	DIC	
2026.42	1964.58	61.84	2088.26	
2039.49	1977.81	61.68	2101.17	(no interaction)

Here we see that the DIC diagnostic is again reduced, showing that fitting separate UV effects for each country was a good idea.

11.5 Problems with univariate updating Metropolis procedures

One note of caution should be made here. If we look at the diagnostic traces for the parameters, for example β_1, the effect for Belgium, (do this via the **Trajectories** window) we will get the following:



Accuracy Diagnostics

Raftery-Lewis (quantile) : Nhat = (159431,281815)
when q = (0.025,0.975), r = 0.005 and s = 0.95
Brooks-Draper (mean) : Nhat = 76991512
when k = 2 sigfigs and alpha = 0.05

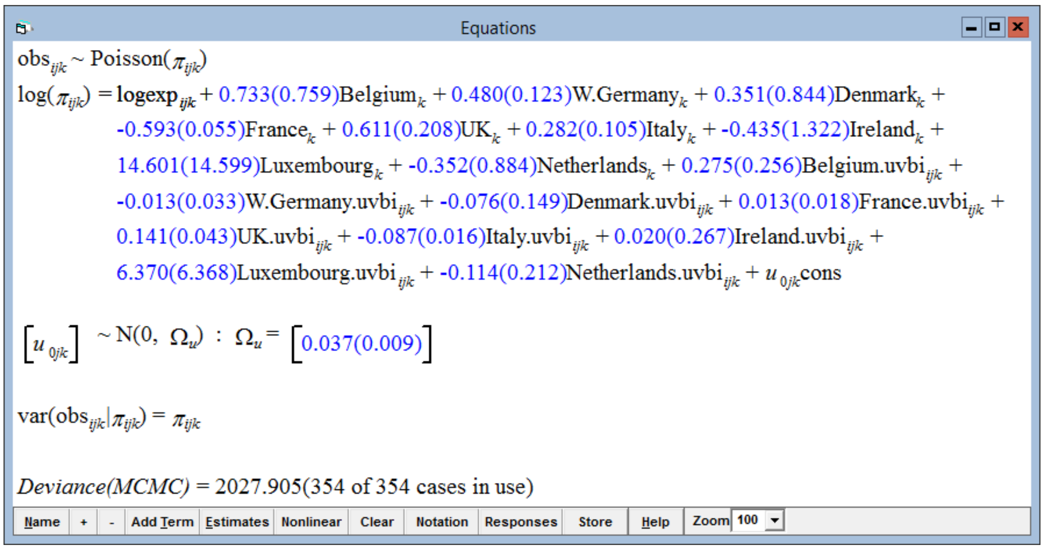
Summary Statistics

param name : β_1 posterior mean = 0.762 (0.100) SD = 0.676 mode = 0.579
quantiles : 2.5% = -0.437, 5% = -0.285, 50% = 0.715, 95% = 1.926, 97.5% = 2.229
50000 actual iterations storing every iteration. Effective Sample Size (ESS) = 40.

As we can see from the ACF graph this trace is very highly correlated and there is no way that we can be confident of the estimates that the MCMC method has produced. All the Poisson models in this chapter can also be fitted in WinBUGS, which will use the Gamerman method described in the last chapter instead of the Metropolis sampler used here. Although generally the Gamerman method produces less correlated chains, for this model the chains produced are also highly correlated. For brevity we omit the WinBUGS analysis here.

A reason for the high autocorrelations is that the parameters in the model are themselves highly correlated. We will revisit this example in chapter 23 to see if we can reparameterise the model to improve the mixing.

We ran the same model for 500,000 iterations using a thinning factor of 10 and got the following estimates:



$\text{obs}_{ijk} \sim \text{Poisson}(\pi_{ijk})$

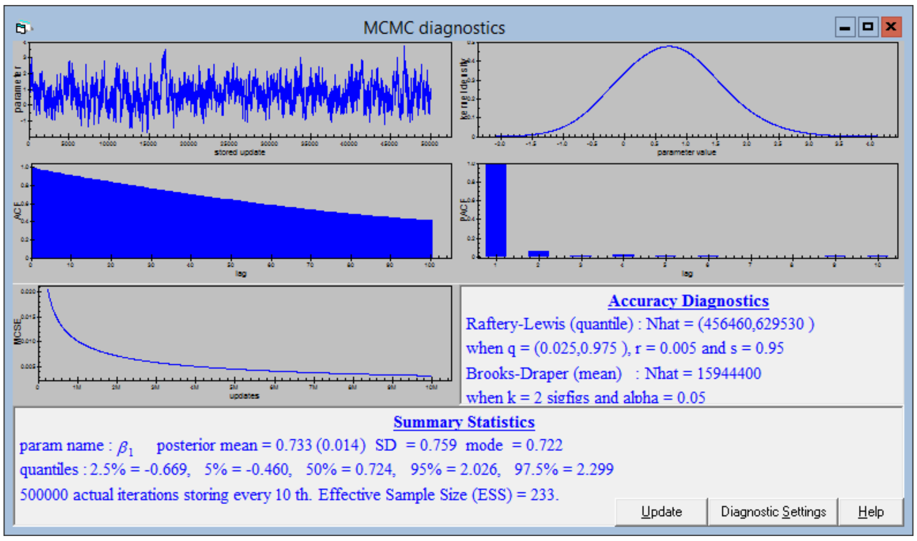
$\log(\pi_{ijk}) = \text{logexp}_{ijk} + 0.733(0.759)\text{Belgium}_k + 0.480(0.123)\text{W.Germany}_k + 0.351(0.844)\text{Denmark}_k + -0.593(0.055)\text{France}_k + 0.611(0.208)\text{UK}_k + 0.282(0.105)\text{Italy}_k + -0.435(1.322)\text{Ireland}_k + 14.601(14.599)\text{Luxembourg}_k + -0.352(0.884)\text{Netherlands}_k + 0.275(0.256)\text{Belgium.uvbi}_{ijk} + -0.013(0.033)\text{W.Germany.uvbi}_{ijk} + -0.076(0.149)\text{Denmark.uvbi}_{ijk} + 0.013(0.018)\text{France.uvbi}_{ijk} + 0.141(0.043)\text{UK.uvbi}_{ijk} + -0.087(0.016)\text{Italy.uvbi}_{ijk} + 0.020(0.267)\text{Ireland.uvbi}_{ijk} + 6.370(6.368)\text{Luxembourg.uvbi}_{ijk} + -0.114(0.212)\text{Netherlands.uvbi}_{ijk} + u_{0jk}\text{cons}$

$\left[u_{0jk}\right] \sim \text{N}(0,\ \Omega_u) \ : \ \Omega_u = \left[0.037(0.009)\right]$

$\text{var}(\text{obs}_{ijk}|\pi_{ijk}) = \pi_{ijk}$

Deviance(MCMC) = 2027.905(354 of 354 cases in use)

These estimates are fairly similar to those found in the User's Guide to MLwiN using quasi-likelihood methods, which is reassuring. If we look in particular at the parameter trace for β_1 we see:



Accuracy Diagnostics

Raftery-Lewis (quantile) : Nhat = (456460,629530)
when q = (0.025,0.975), r = 0.005 and s = 0.95
Brooks-Draper (mean) : Nhat = 15944400
when k = 2 sigfigs and alpha = 0.05

Summary Statistics

param name : β_1 posterior mean = 0.733 (0.014) SD = 0.759 mode = 0.722
quantiles : 2.5% = -0.669, 5% = -0.460, 50% = 0.724, 95% = 2.026, 97.5% = 2.299
500000 actual iterations storing every 10 th. Effective Sample Size (ESS) = 233.

which is a great improvement. The DIC diagnostic is also affected by running the chain for longer here we see:

Dbar	D(thetabar)	pD	DIC	
2027.91	1964.83	63.07	2090.98	(after 500,000)
2026.42	1964.58	61.84	2088.26	(after 50,000)

so our model with interactions is marginally worse than we thought after 50,000 iterations.

Chapter learning outcomes

- ⋆ How to fit Poisson response models.
- ⋆ How to fit an offset term in MLwiN.
- ⋆ How to fit main effects and interactions in Poisson models.

Chapter 12

Unordered Categorical Responses

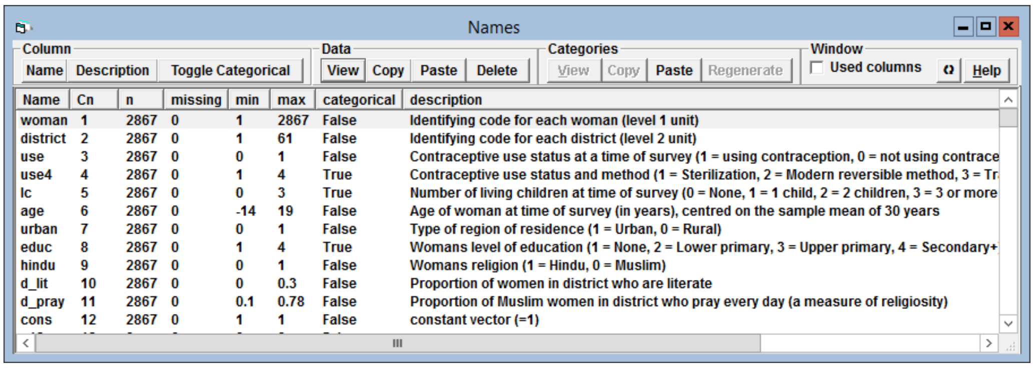
In Chapter 10, we considered fitting models to datasets where the response variable took two possible states, which were classed numerically as 0 and 1. In the example we looked at the two states were whether or not a woman uses contraception in a dataset of Bangladeshi women who took part in the 1988 Fertility survey. The case of binary responses is in fact the simplest case of two possible families of models that we can use when we have 2 or more possible states or categories for a variable.

In this chapter we will consider the case of unordered categories and in Chapter 13 we will look at the alternative extension of binary responses to ordered categorical models. Generally whether a response should be fitted as ordered or unordered is obvious from its definition, however there are some grey areas. For example in voting datasets where there are more than 2 possible parties that voters can vote for we would typically treat the parties as unordered categories. However an alternative in the UK would be to consider three (ordered) categories: **Conservative**, **Labour** and **Other** where **Other** is assumed to lie between the first two.

In this chapter we are going to consider another subset of the Bangladeshi Fertility survey of 1988 where we know not only whether the individual women used contraception but for the women who did we know the method they used. This subset of the dataset is found in the worksheet **bang.ws**. This dataset has also been investigated in the latest version of the User's Guide to MLwiN and here we will generally consider using MCMC for the same models.

- Select **Open Sample Worksheet** from the **File** menu.
- Select **bang.ws** from the list of worksheets.
- Select **Open**.

The **Names** window will then look as follows:



Name	Cn	n	missing	min	max	categorical	description
woman	1	2867	0	1	2867	False	Identifying code for each woman (level 1 unit)
district	2	2867	0	1	61	False	Identifying code for each district (level 2 unit)
use	3	2867	0	0	1	False	Contraceptive use status at a time of survey (1 = using contraception, 0 = not using contrace
use4	4	2867	0	1	4	True	Contraceptive use status and method (1 = Sterilization, 2 = Modern reversible method, 3 = Tr
lc	5	2867	0	0	3	True	Number of living children at time of survey (0 = None, 1 = 1 child, 2 = 2 children, 3 = 3 or more
age	6	2867	0	-14	19	False	Age of woman at time of survey (in years), centred on the sample mean of 30 years
urban	7	2867	0	0	1	False	Type of region of residence (1 = Urban, 0 = Rural)
educ	8	2867	0	1	4	True	Womans level of education (1 = None, 2 = Lower primary, 3 = Upper primary, 4 = Secondary+
hindu	9	2867	0	0	1	False	Womans religion (1 = Hindu, 0 = Muslim)
d_lit	10	2867	0	0	0.3	False	Proportion of women in district who are literate
d_pray	11	2867	0	0.1	0.78	False	Proportion of Muslim women in district who pray every day (a measure of religiosity)
cons	12	2867	0	1	1	False	constant vector (=1)

In this subset of the dataset we have in the column **use** whether or not a woman uses contraception and then in the column **use4** a categorisation of the type of contraception used. The categories for the variable are

1. Sterilization of the woman or her partner.
2. Modern reversible methods.
3. Traditional methods.
4. Not using contraception.

We will be treating the four categories as unordered although there is a vague ordering in terms of effectiveness of preventing pregnancy of the four categories. This subset of the Bangladeshi dataset shares the majority of its predictor variables with the subset used in Chapter 10 and we will here only consider a few of these variables in our modelling.

To motivate multinomial modelling we will firstly look at the distribution of the **use4** variable via the **Tabulate** window.

- Select **Tabulate** from the **Basic Statistics** menu.
- Click on the **Percentage of column totals** tick box.
- Select **use4** from the list next to **Columns**.
- Click on the **Tabulate** button.

This will run the **TABUlate** command and give the following results:

	1	2	3	4	TOTALS
N	302	555	282	1728	2867
%	10.5	19.4	9.8	60.3	100.0

Here we see that 60.3% of the women do not use contraception, whilst of those that do roughly half use modern reversible methods.

12.1 Fitting a first single-level multinomial model

The simplest multinomial model we can fit would simply encapsulate the information in the above tabulation. If we let the probability of an individual i having response r be $\pi_i^{(r)} = \Pr(y_i = r)$, then following on from the binary response case we assume that one of the categories is the reference category (0 in the binary model). Then in the multinomial logistic model we construct a model that compares each of the categories with the baseline category. For example with t categories and category t as the baseline:

$$\log\left(\frac{\pi_i^{(r)}}{\pi_i^{(t)}}\right) = (X_iB)^{(r)}, \qquad r = 1, \ldots, t-1$$

We therefore have $t-1$ equations each of which is equivalent to a binary response model comparing the probabilities of each category with the base category. Here $X^{(r)}$ is a set of predictor variables that may be common for each equation or may be different. The corresponding coefficients $\beta^{(r)}$ are unique coefficients for the rth equation. Although the $\beta^{(r)}$ are themselves not particularly meaningful to interpret, it is often preferable to calculate (for given predictor values, X_i) the predicted probabilities for the t categories. These are constructed as follows:

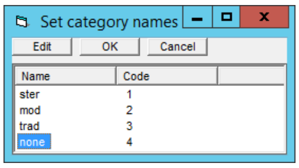
$$\hat{\pi}_i^{(r)} = \frac{\exp((X_i\hat{B})^{(r)})}{1 + \sum_{k=1}^{t-1} \exp((X_i\hat{B})^{(k)})}, \quad r = 1, ..., t-1, \quad \hat{\pi}_i^{(t)} = 1 - \sum_{k=1}^{t-1} \pi_i^{(k)}$$

We will now construct a simple model that will recover the probabilities that the **TABUlate** command gave us.

We will firstly need to set up the category names for the **use4** variable.

- Select **Names** from the **Data Manipulation** menu.
- Click on **use4** from the list and then the **View** button in the categories section.
- For the categories 1 through to 4 type in turn **ster**, **mod**, **trad** and **none**.

The window should look as follows:



Name	Code
ster	1
mod	2
trad	3
none	4

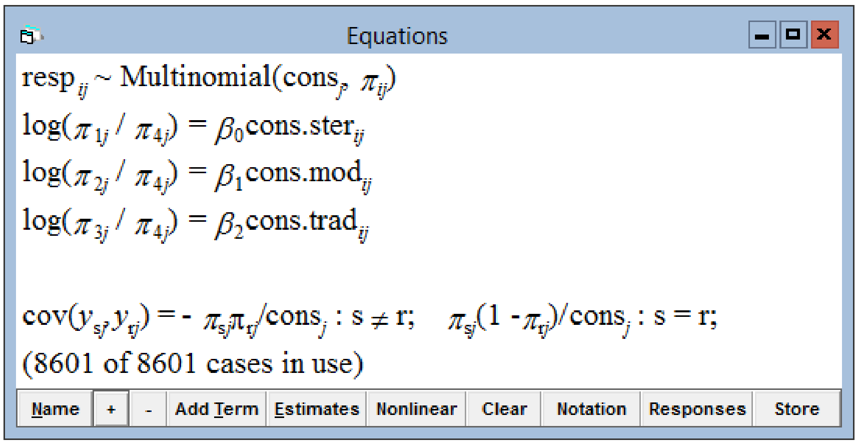
We will next click on the **OK** button and then set up the model in the **Equations** window as follows:

- Select **Equations** from the **Model** menu.
- Click on the red y and select **use4** as the y variable.
- Select **1-i** for the number (N) of levels.
- Select **woman** as the **Level 1(i)** identifier.
- Click on the **Done** button.
- Click on the **N** and from the list of distributions that appears scroll down and select **Multinomial**.
- Change the reference category from **ster** to **none** and click on the **Done** button.

Next we need to define the denominator and the intercepts for each category:

- Click on the red n_j to set the denominator.
- Select **cons** from the list that appears and click on the **Done** button.
- Click on the **Add Term** button.
- Select **cons** from the variable list.
- Click on the **add Separate coefficients** button.

The **Equations** window should now look (after pressing the + button twice) as follows:



$\text{resp}_{ij} \sim \text{Multinomial}(\text{cons}_j, \pi_{ij})$

$\log(\pi_{1j} / \pi_{4j}) = \beta_0 \text{cons.ster}_{ij}$

$\log(\pi_{2j} / \pi_{4j}) = \beta_1 \text{cons.mod}_{ij}$

$\log(\pi_{3j} / \pi_{4j}) = \beta_2 \text{cons.trad}_{ij}$

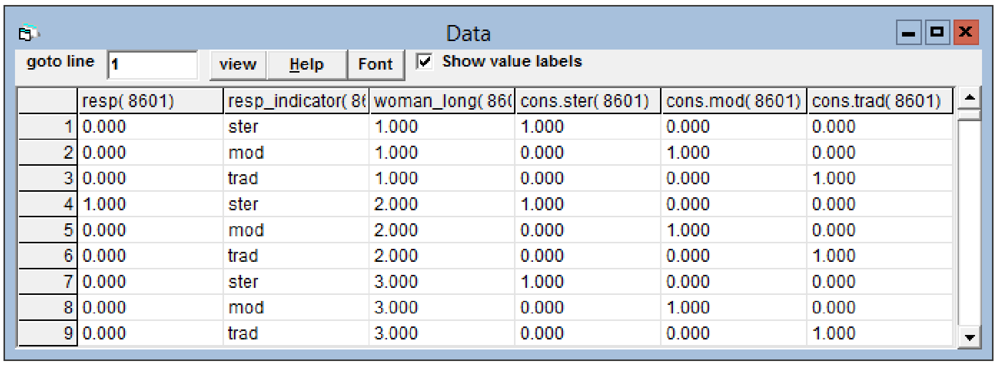
$\text{cov}(y_{sj}, y_{rj}) = -\ \pi_{sj}\pi_{rj}/\text{cons}_j : s \neq r; \quad \pi_{sj}(1 - \pi_{rj})/\text{cons}_j : s = r;$

(8601 of 8601 cases in use)

Here we see the three equations that relate the four categories. When selecting multinomial modelling, MLwiN performs many data manipulation operations behind the scenes to construct an expanded response variable and to expand the predictors accordingly. This is a similar process to the expansion in multivariate models that we will describe in Chapter 18.

- Select **View or Edit Data** from the **Data Manipulation** menu.
- Click on the **View** button.
- Select the following 6 columns: **resp**, **resp_indicator**, **woman_long**, **cons.ster**, **cons.mod**, **cons.trad**.
- Click on the **OK** button.

The **Data** window looks as follows:



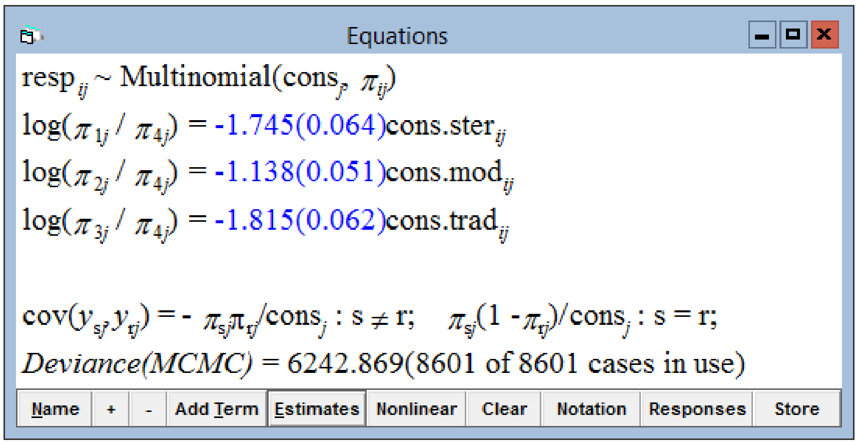
	resp(8601)	resp_indicator(86	woman_long(86(	cons.ster(8601)	cons.mod(8601)	cons.trad(8601)
1	0.000	ster	1.000	1.000	0.000	0.000
2	0.000	mod	1.000	0.000	1.000	0.000
3	0.000	trad	1.000	0.000	0.000	1.000
4	1.000	ster	2.000	1.000	0.000	0.000
5	0.000	mod	2.000	0.000	1.000	0.000
6	0.000	trad	2.000	0.000	0.000	1.000
7	0.000	ster	3.000	1.000	0.000	0.000
8	0.000	mod	3.000	0.000	1.000	0.000
9	0.000	trad	3.000	0.000	0.000	1.000

Here we see that for each observation in the original dataset we have 3 records in the new dataset, one for each category (excluding the base category). The **resp** column transforms the response from a number between 1 and 4 to three binary indicators. The **resp_indicator** column simply identifies each indicator with a category. The **woman_long** column identifies the individual woman for each record and the remaining 3 columns give the three intercept terms.

If we now wish to fit this first model using MCMC we need to do the following:

- Select **Equations** from the **Model** menu.
- Click on the **Nonlinear** button.
- On the window that appears click on the **Use Defaults** and **Done** buttons.
- Click on the **Start** button (to run using IGLS 1st order MQL).
- Select **MCMC** from the **Estimation** menu.
- Click on the **Start** button.

The model will now be run using MCMC and after a few minutes we should get the following estimates (after pressing the estimates button twice):



$\text{resp}_{ij} \sim \text{Multinomial}(\text{cons}_j, \pi_{ij})$

$\log(\pi_{1j} / \pi_{4j}) = -1.745(0.064)\text{cons.ster}_{ij}$

$\log(\pi_{2j} / \pi_{4j}) = -1.138(0.051)\text{cons.mod}_{ij}$

$\log(\pi_{3j} / \pi_{4j}) = -1.815(0.062)\text{cons.trad}_{ij}$

$\text{cov}(y_{sj}, y_{rj}) = -\pi_{sj}\pi_{rj}/\text{cons}_j : s \neq r; \quad \pi_{sj}(1 - \pi_{rj})/\text{cons}_j : s = r;$

Deviance(MCMC) = 6242.869(8601 of 8601 cases in use)

Here we see that the three intercepts are all negative which shows that all these three categories are less likely than the base category. To convert to the underlying probabilities we can use the equations earlier and the **CALC** command in the **Command Interface** window available from the **Data Manipulation** menu. For the probability of using sterilisation predicted by the model we have:

```
▶ CALC EXPO(-1.745) / (1 + EXPO(-1.745) + EXPO(-1.138) +
  EXPO(-1.815))
```

which gives 0.10534. Similarly for the modern and traditional methods we have

```
▶ CALC EXPO(-1.138) / (1+EXPO(-1.745) + EXPO(-1.138) +
  EXPO(-1.815))
```

and

```
▶ CALC EXPO(-1.815) / (1+EXPO(-1.745) + EXPO(-1.138) +
  EXPO(-1.815))
```

which give 0.19329 and 0.098217 respectively. So we see that the model returns exactly (apart from rounding errors) the probabilities that we got through the **TABUlate** command earlier. Of course this model is the simplest we could fit and now we can add more predictors and levels to the model.

We will firstly however consider the DIC diagnostic, which we can also use for multinomial models. For multinomial models the deviance formula is

$$D = -2 \sum_i \sum_{j=1}^{t} [I(y_i = j)\hat{\pi}_i^{(j)}]$$

where $I()$ is an indicator function which returns 1 if the condition is satisfied i.e. if individual i uses method j. $\hat{\pi}_i^{(j)}$ is the estimated probability of being in category j for individual i. This means that the MCMC engine in MLwiN calculates the estimated probabilities as part of the DIC diagnostic command.

- Select **MCMC/DIC diagnostic** from the **Model** menu.

The output can be seen below:

Dbar	D(thetabar)	pD	DIC
6242.87	6239.80	3.07	6245.94

Here we can again see that the DIC picks up the correct degrees of freedom. We can now consider adding in predictor variables and see if they produce a better model via the DIC diagnostic.

12.2 Adding predictor variables

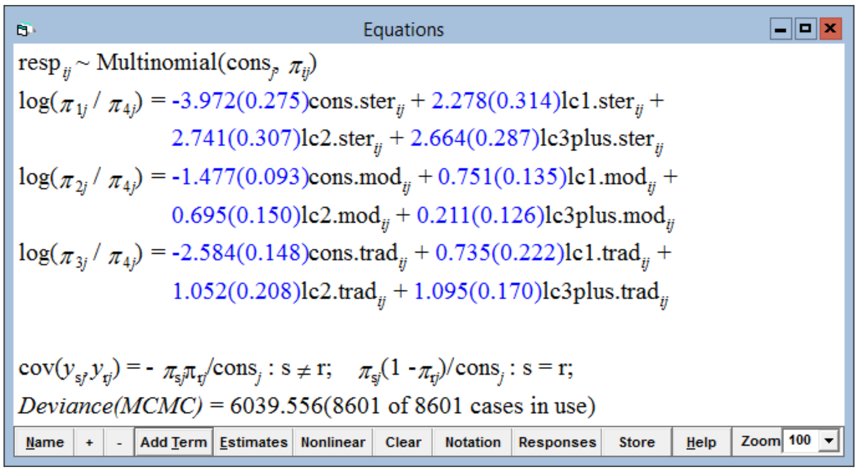
In Chapter 10 we discovered that, for the subset of the Bangladeshi dataset used there, the number of living children a woman had increased her chances of using contraception. If we now consider this predictor in the multinomial example, we can look at the effect of the number of children on each of the three types of contraceptive use.

- Change **Estimation method** to **IGLS**.
- Click on the **Add Term** button on the **Equations** window.
- Select **lc** in the **variable** list and keep **lc0** as the reference category.
- Click on the **add Separate coefficients** button.

As there are four categories of number of living children and four categories of contraceptive use, the above commands will add $(4-1)\times(4-1) = 9$ fixed effect parameters. To run the model using MCMC we now need to do the following:

- Click on the **Start** button (to run using IGLS 1st order MQL).
- Select **MCMC** from the **Estimation** menu.
- Click on the **Start** button.

After a few minutes the model runs and gives the following estimates:



$\text{resp}_{ij} \sim \text{Multinomial}(\text{cons}_j, \pi_{ij})$

$\log(\pi_{1j}/\pi_{4j}) = -3.972(0.275)\text{cons.ster}_{ij} + 2.278(0.314)\text{lc1.ster}_{ij} + 2.741(0.307)\text{lc2.ster}_{ij} + 2.664(0.287)\text{lc3plus.ster}_{ij}$

$\log(\pi_{2j}/\pi_{4j}) = -1.477(0.093)\text{cons.mod}_{ij} + 0.751(0.135)\text{lc1.mod}_{ij} + 0.695(0.150)\text{lc2.mod}_{ij} + 0.211(0.126)\text{lc3plus.mod}_{ij}$

$\log(\pi_{3j}/\pi_{4j}) = -2.584(0.148)\text{cons.trad}_{ij} + 0.735(0.222)\text{lc1.trad}_{ij} + 1.052(0.208)\text{lc2.trad}_{ij} + 1.095(0.170)\text{lc3plus.trad}_{ij}$

$\text{cov}(y_{sj}, y_{rj}) = -\pi_{sj}\pi_{rj}/\text{cons}_j : s \neq r; \quad \pi_{sj}(1-\pi_{rj})/\text{cons}_j : s = r;$

Deviance(MCMC) = 6039.556(8601 of 8601 cases in use)

Here we see that for all three types of contraception there is greater probability of usage if the woman has had some children although the pattern is not the same for all methods. For the modern (reversible) method there is greater probability of usage (as opposed to non-usage) in women with 1 or 2 living children and less probability for women with 3 or more children and the least probability in women with no children. In contrast the probability of using traditional methods increases with the number of living children.

The User's Guide to MLwiN gives a macro for calculating the probabilities for this particular model which essentially involves using a similar formula that we used for the last model, so for example to calculate the conditional probability of a woman with no living children using traditional methods we have the following calculation

```
▶ CALC EXPO(-2.584) / (1+EXPO(-3.972) + EXPO(-1.477) +
  EXPO(-2.584))
```

which gives a probability of 0.0571. In contrast for a woman with 2 children the calculation is as follows:

```
▶ CALC EXPO(-2.584+1.052) / (1+EXPO(-3.972+2.741) +
  EXPO(-1.477+0.695) + EXPO(-2.584+1.052))
```

which gives a probability of 0.110.

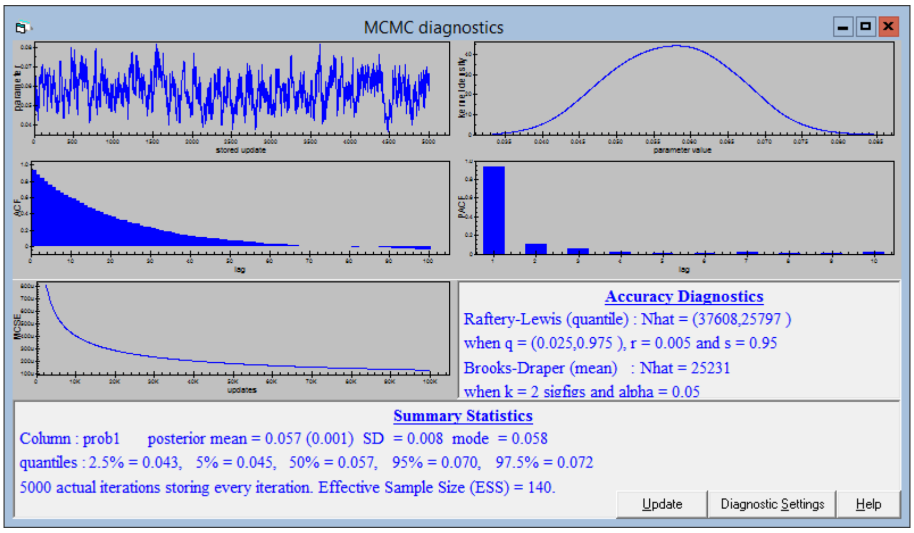
As shown in the User's Guide to MLwiN the **TABUlate** command can be used to construct the counts and probabilities of each of the 16 categories in a 2 way table of contraceptive use and number of living children. Then by considering the particular row corresponding to the number of living children we can construct the probabilities given by the model.

12.3 Interval estimates for conditional probabilities

One of the advantages of MCMC methods is that as they are simulation-based approaches we can construct interval estimates from the chains of values. We can also, as we saw in Chapter 4, calculate interval estimates for any derived parameters as well, for example the above conditional probabilities. MLwiN stores the chains of parameter estimates stacked in column **c1090** and so we firstly need to split this column into 13 columns, 1 for each of the 12 fixed effects plus one which contains a level 1 random effect that is constrained to 1 at each iteration. Then the above calculations are carried out on the columns rather than the point estimates. The following commands can be typed in the **Command interface** window:

```
▶ code 13 1 5000 c300
▶ split c1090 c300 c301-c313
▶ calc c314 = expo(c303) / (1+expo(c301) + expo(c302) +
  expo(c303))
▶ calc c315 = expo(c303+c311)/ (1 + expo(c301+c305) +
  expo(c302+c308) + expo(c303+c311))
▶ name c314 'prob1' c315 'prob2'
```

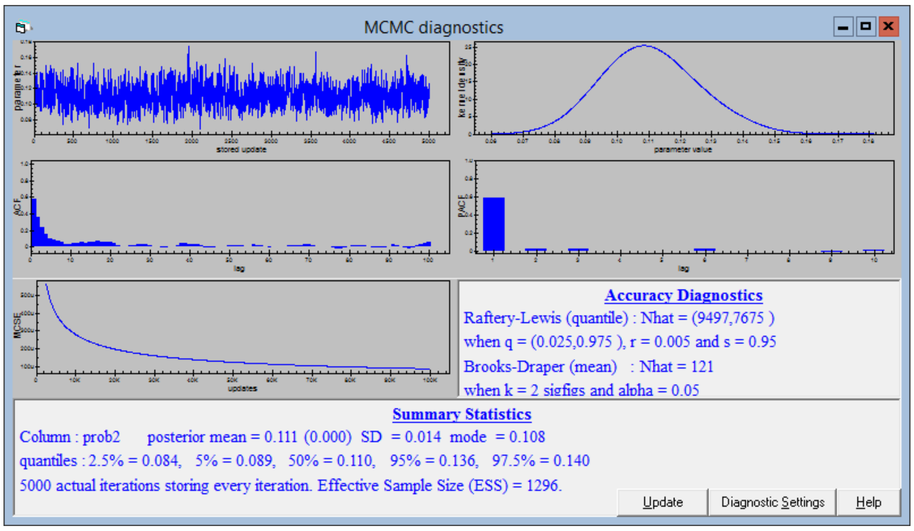
This will put chains of values for the two probabilities in columns **c314** and **c315** and we can look at them using the **Column Diagnostics** window available from the **Basic Statistics** menu as follows for the first probability (of using traditional methods given the woman has no children):



Accuracy Diagnostics
Raftery-Lewis (quantile) : Nhat = (37608,25797)
when q = (0.025,0.975), r = 0.005 and s = 0.95
Brooks-Draper (mean) : Nhat = 25231
when k = 2 sigfigs and alpha = 0.05

Summary Statistics
Column : prob1 posterior mean = 0.057 (0.001) SD = 0.008 mode = 0.058
quantiles : 2.5% = 0.043, 5% = 0.045, 50% = 0.057, 95% = 0.070, 97.5% = 0.072
5000 actual iterations storing every iteration. Effective Sample Size (ESS) = 140.

So here we have a 95% credible interval for the probability of between 0.043 and 0.072. Similarly for the probability of using traditional methods given the woman has 2 children we get an interval of (0.084,0.140) as shown below:



Accuracy Diagnostics
Raftery-Lewis (quantile) : Nhat = (9497,7675)
when q = (0.025,0.975), r = 0.005 and s = 0.95
Brooks-Draper (mean) : Nhat = 121
when k = 2 sigfigs and alpha = 0.05

Summary Statistics
Column : prob2 posterior mean = 0.111 (0.000) SD = 0.014 mode = 0.108
quantiles : 2.5% = 0.084, 5% = 0.089, 50% = 0.110, 95% = 0.136, 97.5% = 0.140
5000 actual iterations storing every iteration. Effective Sample Size (ESS) = 1296.

Finally if we now look at the DIC diagnostic we get the following:

Dbar	D(thetabar)	pD	DIC	
6039.56	6027.92	11.64	6051.20	(current model)
6242.87	6239.80	3.07	6245.94	(without living children predictors)

Here we see that the DIC diagnostic picks up (approximately) that we have added 9 parameters (11.64 – 3.07) and the DIC value has reduced by nearly 200 suggesting this is a much better model. We could of course now consider some of the other predictors as we did in Chapter 10 but here we will move on to consider adding random effects.

12.4 Adding district level random effects

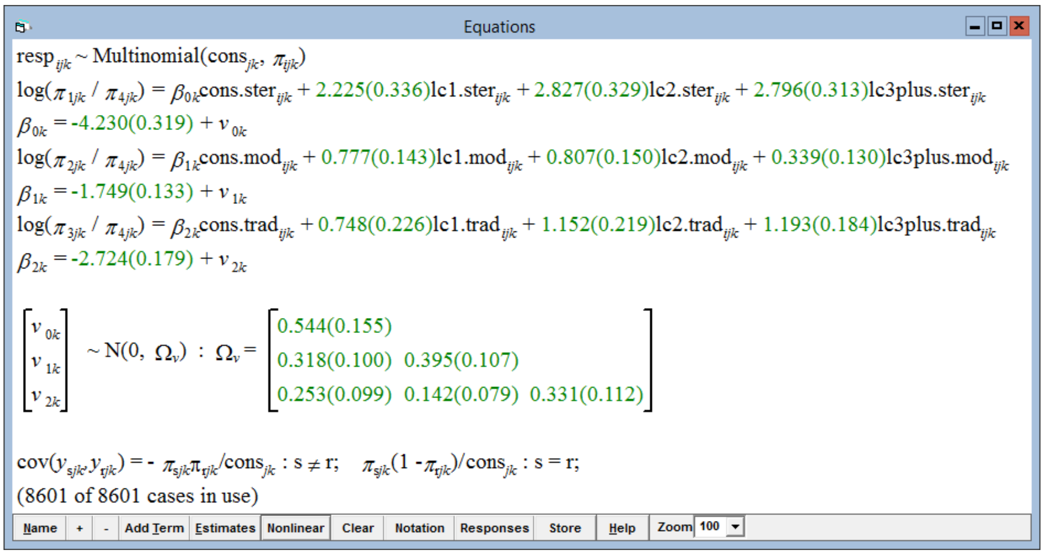
The Bangladeshi dataset contains two levels with women nested within districts and so far we have assumed that the probability of using the various types of contraception will be constant across districts. As we have 60 districts we will treat the differences between districts as random effects. As there are 3 contraceptive methods this will result in 3 sets of random effects.

- Change **Estimation method** to **IGLS**.
- Select **Equations** from the **Model** menu.
- Click on the response variable to bring up the **Y variable** window.
- Select **3-ijk** for the **N levels** indicator.
- Select **district** as the **level 3(k)** identifiers.
- Click on the **Done** button.
- Click on **cons.ster** and select the **k(district_long)** tickbox.
- Click on the **Done** button.
- Click on **cons.mod** and select the **k(district_long)** tickbox.
- Click on the **Done** button.
- Click on **cons.trad** and select the **k(district_long)** tickbox.
- Click on the **Done** button.

In the User's Guide to MLwiN it was pointed out that the estimates from 1st order MQL estimation were often severely biased for multilevel multinomial models so here we will move straight to 2nd order PQL estimation to get starting values for MCMC.

- Click on the **Nonlinear** button on the **Equations** window.
- Select **2nd Order** and **PQL** from the options that appear.
- Click on the **Done** button.
- Click on the **Start** button.

After 11 iterations the model will converge to give the following estimates:



$\text{resp}_{ijk} \sim \text{Multinomial}(\text{cons}_{jk}, \pi_{ijk})$

$\log(\pi_{1jk} / \pi_{4jk}) = \beta_{0k}\text{cons.ster}_{ijk} + 2.225(0.336)\text{lc1.ster}_{ijk} + 2.827(0.329)\text{lc2.ster}_{ijk} + 2.796(0.313)\text{lc3plus.ster}_{ijk}$

$\beta_{0k} = -4.230(0.319) + v_{0k}$

$\log(\pi_{2jk} / \pi_{4jk}) = \beta_{1k}\text{cons.mod}_{ijk} + 0.777(0.143)\text{lc1.mod}_{ijk} + 0.807(0.150)\text{lc2.mod}_{ijk} + 0.339(0.130)\text{lc3plus.mod}_{ijk}$

$\beta_{1k} = -1.749(0.133) + v_{1k}$

$\log(\pi_{3jk} / \pi_{4jk}) = \beta_{2k}\text{cons.trad}_{ijk} + 0.748(0.226)\text{lc1.trad}_{ijk} + 1.152(0.219)\text{lc2.trad}_{ijk} + 1.193(0.184)\text{lc3plus.trad}_{ijk}$

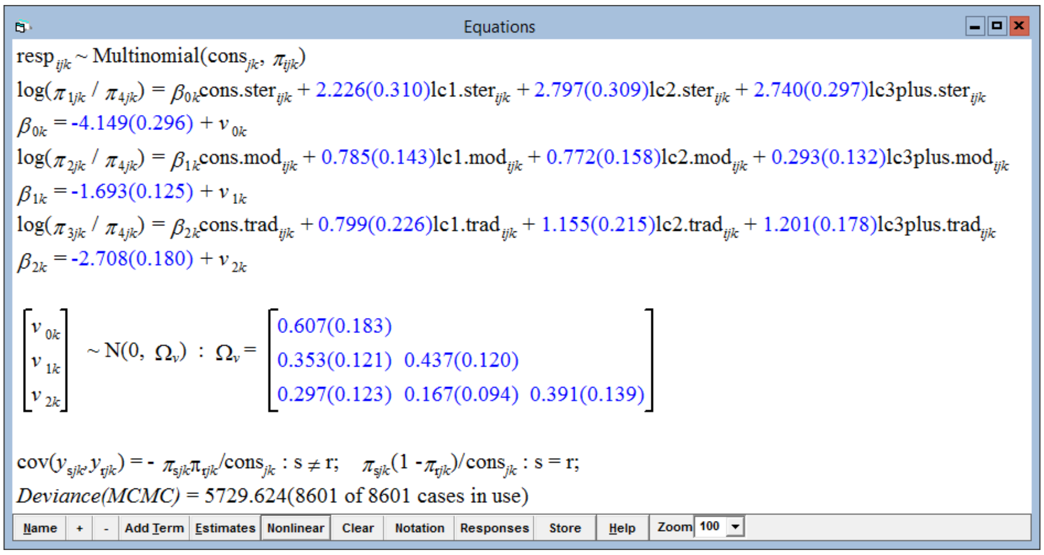
$\beta_{2k} = -2.724(0.179) + v_{2k}$

$$\begin{bmatrix} v_{0k} \\ v_{1k} \\ v_{2k} \end{bmatrix} \sim \text{N}(0,\ \Omega_v) \ : \ \Omega_v = \begin{bmatrix} 0.544(0.155) & & \\ 0.318(0.100) & 0.395(0.107) & \\ 0.253(0.099) & 0.142(0.079) & 0.331(0.112) \end{bmatrix}$$

$\text{cov}(y_{sjk}, y_{rjk}) = -\ \pi_{sjk}\pi_{rjk}/\text{cons}_{jk} : s \neq r; \quad \pi_{sjk}(1 - \pi_{rjk})/\text{cons}_{jk} : s = r;$

(8601 of 8601 cases in use)

If we now change estimation method to MCMC and press the **Start** button we can compare methods:



$\text{resp}_{ijk} \sim \text{Multinomial}(\text{cons}_{jk}, \pi_{ijk})$

$\log(\pi_{1jk} / \pi_{4jk}) = \beta_{0k}\text{cons.ster}_{ijk} + 2.226(0.310)\text{lc1.ster}_{ijk} + 2.797(0.309)\text{lc2.ster}_{ijk} + 2.740(0.297)\text{lc3plus.ster}_{ijk}$

$\beta_{0k} = -4.149(0.296) + v_{0k}$

$\log(\pi_{2jk} / \pi_{4jk}) = \beta_{1k}\text{cons.mod}_{ijk} + 0.785(0.143)\text{lc1.mod}_{ijk} + 0.772(0.158)\text{lc2.mod}_{ijk} + 0.293(0.132)\text{lc3plus.mod}_{ijk}$

$\beta_{1k} = -1.693(0.125) + v_{1k}$

$\log(\pi_{3jk} / \pi_{4jk}) = \beta_{2k}\text{cons.trad}_{ijk} + 0.799(0.226)\text{lc1.trad}_{ijk} + 1.155(0.215)\text{lc2.trad}_{ijk} + 1.201(0.178)\text{lc3plus.trad}_{ijk}$

$\beta_{2k} = -2.708(0.180) + v_{2k}$

$$\begin{bmatrix} v_{0k} \\ v_{1k} \\ v_{2k} \end{bmatrix} \sim \text{N}(0,\ \Omega_v) \ : \ \Omega_v = \begin{bmatrix} 0.607(0.183) & & \\ 0.353(0.121) & 0.437(0.120) & \\ 0.297(0.123) & 0.167(0.094) & 0.391(0.139) \end{bmatrix}$$

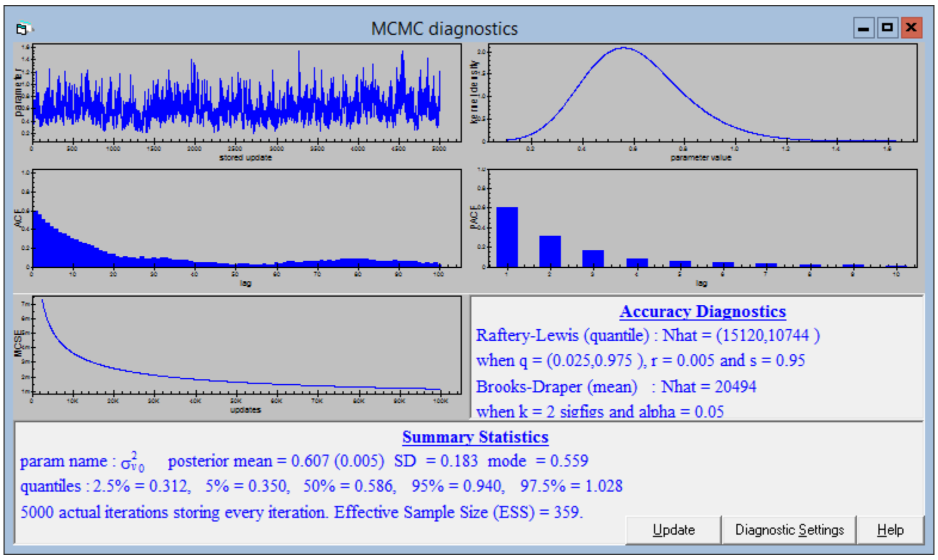
$\text{cov}(y_{sjk}, y_{rjk}) = -\ \pi_{sjk}\pi_{rjk}/\text{cons}_{jk} : s \neq r; \quad \pi_{sjk}(1 - \pi_{rjk})/\text{cons}_{jk} : s = r;$

Deviance(MCMC) = 5729.624(8601 of 8601 cases in use)

Here we see that as is often the case the MCMC estimation method gives larger (in magnitude) estimates. If we now look at the chain for the first variance, σ^2_{v0}, we can investigate this further.

- Select **Trajectories** from the **Model** menu.
- Click on the **Select** button and highlight **district_long : cons.ster/cons.ster**
- Click on the **Done** button.
- Click on the trajectory plot and the **Yes** box that appears.

The MCMC diagnostics appear as follows:



Accuracy Diagnostics
Raftery-Lewis (quantile) : Nhat = (15120,10744)
when q = (0.025,0.975), r = 0.005 and s = 0.95
Brooks-Draper (mean) : Nhat = 20494
when k = 2 sigfigs and alpha = 0.05

Summary Statistics
param name : σ^2_{v0} posterior mean = 0.607 (0.005) SD = 0.183 mode = 0.559
quantiles : 2.5% = 0.312, 5% = 0.350, 50% = 0.586, 95% = 0.940, 97.5% = 1.028
5000 actual iterations storing every iteration. Effective Sample Size (ESS) = 359.

Here we see that in fact the posterior mode estimate of 0.607 agrees favourably with the 2nd order PQL estimate of 0.544 suggesting that the differences here are mainly due to the fact that we were originally comparing mean with mode (ML) estimates. It is also noticeable that we should in fact have run for longer than the 5,000 iterations. In fact some of the fixed effect traces suggest a much longer run.

Considering the model we have fitted in more detail we can firstly use the DIC diagnostic to compare this model with the last model. To do this we select **MCMC/DIC diagnostic** from the **Model** menu, which gives:

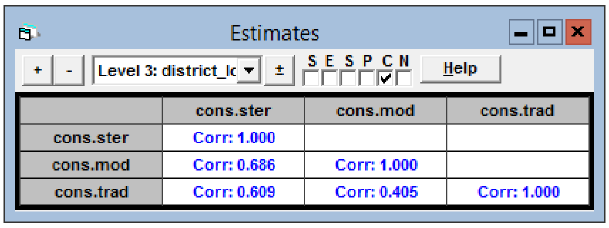
Dbar	D(thetabar)	pD	DIC	
5729.62	5624.77	104.85	5834.47	(Model with random effects)
6039.56	6027.92	11.64	6051.20	(Model without random effects)

Here we see that the DIC diagnostic has again been reduced, this time by over 200 suggesting that the random effects model is significantly better. The 180 random effects that we have added have contributed $106.41 - 11.95 = 94.46$ effective parameters.

Looking at the parameter estimates we see that the three covariances are positive and we can look at the correlations between the sets of random effects as follows:

- Select **Estimate Tables** from the **Model** menu.
- Select **Level 3: district_long** from the drop-down list.
- Select C and unselect S, E, S and P at the top right of the window.

You should now have the following correlation matrix.



	cons.ster	cons.mod	cons.trad
cons.ster	Corr: 1.000		
cons.mod	Corr: 0.686	Corr: 1.000	
cons.trad	Corr: 0.609	Corr: 0.405	Corr: 1.000

Here we see that there are fairly strong correlations between the three sets of random effects. This means that generally in districts with higher sterilisation probabilities, there are also higher probabilities of modern and traditional methods as well. More succinctly the main differentiation between districts is that there are some districts with high contraceptive use and others with low contraceptive use. However, the type of use does not exhibit a particularly strong pattern such as, for example, higher probabilities of traditional methods in districts with lower rates of usage of the other two methods.

Care must be taken when interpreting these correlations as the numbers in the base category will affect all three random effects simultaneously and so it is to be expected that there will be some positive correlation.

In the User's Guide to MLwiN the residuals produced by this model along with plots are considered. Similar plots can be produced for the residual estimates produced by the MCMC methods and we will leave these as an exercise for the user.

Chapter learning outcomes

⋆ How to fit models to unordered categorical responses

⋆ How to extend such models to include random effects

⋆ How to calculate point and interval estimates of conditional probabilities

⋆ How to compare MCMC estimates with IGLS estimates

Chapter 13

Ordered Categorical Responses

In the last chapter we considered datasets where the response is one of a selection of possible alternatives, where the alternatives do not have a particular ordering. In this chapter we will consider the case where the alternatives do have a natural ordering and so we can build this ordering into our model. It is of course possible to use the models in the last chapter with ordered data as in some way they can be considered a special case of categorical data and the general multinomial models in the last chapter will fit any types of categorical data.

Ordered categorical responses occur in many fields. In education, exam marks are often graded on a scale, for example A, B, C, D, E, and U with A representing the best marks and U representing the worst marks (failures). Other examples are survey questionnaires where individuals have to give their level of agreement with a statement, typically ranging from 'Strongly Agree' through to 'Strongly disagree' and finally in health where a person's illness/fitness may be classified on a scale.

Another possibility for fitting ordered categorical data is to treat the response variable as continuous and fit Normal response models. This will make more sense the greater the number of categories we have and in fact in education such modelling is often performed on exam marks or grade scores. In the example we will consider in this chapter our response is the mark in an A level (taken at age 18) chemistry exam and we will compare and contrast the approaches of treating this response as continuous and as an ordered response.

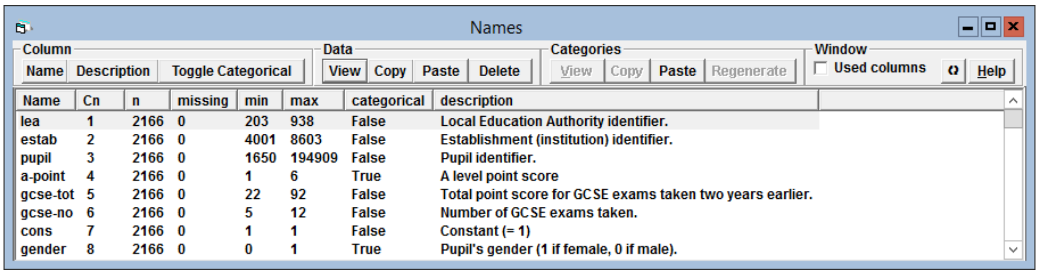
13.1 A level chemistry dataset

The data that we analyse is taken from a larger dataset that contains all the A level exam results taken in England for the period 1994–1997 (Yang &

Woodhouse, 2001) . The subset of the dataset used here is all the chemistry exam results for one examinations board in the year 1997.

- Select **Open sample worksheet** from the **File** menu.
- Select **alevchem.ws** from the list of worksheets.
- Select **Open**.

The **Names** window will then appear as we see below:



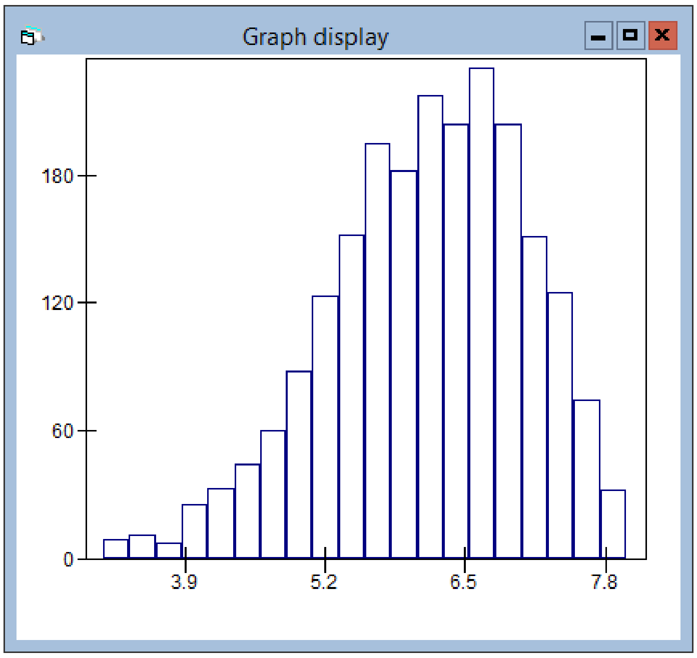
Name	Cn	n	missing	min	max	categorical	description
lea	1	2166	0	203	938	False	Local Education Authority identifier.
estab	2	2166	0	4001	8603	False	Establishment (institution) identifier.
pupil	3	2166	0	1650	194909	False	Pupil identifier.
a-point	4	2166	0	1	6	True	A level point score
gcse-tot	5	2166	0	22	92	False	Total point score for GCSE exams taken two years earlier.
gcse-no	6	2166	0	5	12	False	Number of GCSE exams taken.
cons	7	2166	0	1	1	False	Constant (= 1)
gender	8	2166	0	0	1	True	Pupil's gender (1 if female, 0 if male).

Here we see that there are in fact three levels to the data with 2166 pupils (**pupil**) nested within 219 schools (**estab**) nested within 70 LEAs (**lea**) although we will only consider the first two levels here. The response variable is **a-point** which ranges from 1 (grade F) to 6 (grade A). There are three predictor variables, **gcse-tot** and **gcse-no** give the total points and number of GCSE exams sat at age 16 and finally the sex, **gender**, of each child with a value of 1 representing a girl and 0 a boy. It should be noted that the GCSE system has 8 grades rather than 6 and these range from A* worth 8 points then A worth 7 points through to G worth 1 point. The extra grade A* was introduced for the first time in 1994.

Perhaps one of the first things we can do with the dataset is to construct the average GCSE scores as these would be preferable to the two predictors, total GCSE score and total number of GCSEs. This can be achieved by typing the following commands in the **Command interface** window (available from the **Data Manipulation** menu):

```
▶ CALC c9=c5/c6
```

Here unlike in the User's Guide to MLwiN we do not Normalise our response or predictor variables. The A level score response can only take 6 possible values and so the Normalising transform will only have limited effect. As shown below (by selecting **histogram** from the **Customised graph** window) although the GCSE scores are slightly skewed to the left for interpretation we maintain the raw scores.



We will however centre the predictor around a GCSE average score of 6 (note the actual average score is 6.15). This will make the intercept more meaningful as it will now represent the prediction for a child with an average GCSE score of 6. We will also consider quadratic and cubic coefficients of the predictor by typing the following commands in the **Command interface** window.

```
▶ CALC c9=c9-6
▶ CALC c10=c9^2
▶ CALC c11=c9^3
▶ NAME c9 'gcseav' c10 'gcse^2' c11 'gcse^3'
```

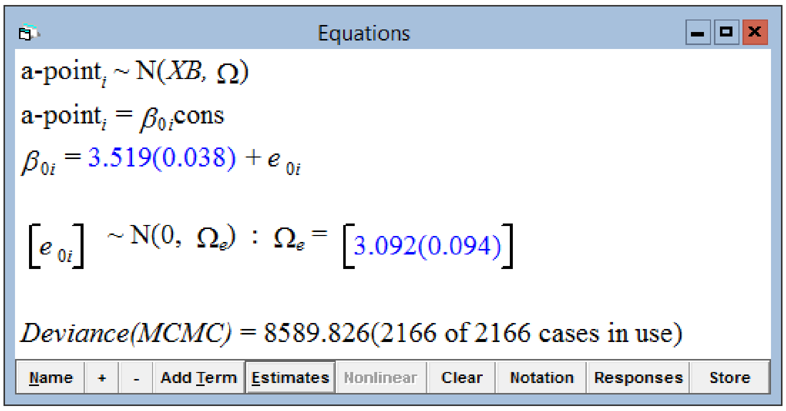
For our response variable we are simply assuming that an F is worth 1 point, and an A is worth 6 points. This would seem rather arbitrary but will lead on to ordered response models which essentially remove this linear incremental approach. The alternative approach of using Normalised scores basically gives an alternative mark to each grade from F to A based on their frequency of occurring and an underlying Normal distribution assumption. Here the scores would be −1.287 for an F, −0.629 for an E, −0.245 for a D, 0.150 for a C, 0.639 for a B and 1.38 for an A. We can see that the gaps between the extremes (A/B and E/F) are larger. We will come back to this when we fit the ordered response models.

13.2 Normal response models

To firstly set up a Normal response model we need to do the following:

- Select **Equations** from the **Model** menu.
- Click on the red y.
- Select **a-point** for the y variable.
- Select **1-i** for the number (N) of levels.
- Select **pupil** as **level 1(i)** and click on the **Done** button.
- Click on the red x_0 and select **cons** from the list.
- Select the **i-pupil** tick box keeping the **Fixed Parameter** box selected.
- Click on the **Done** button.
- Click on the **Start** button.
- Change **Estimation method** to **MCMC**.
- Click on the **Start** button.

After running for 5,000 iterations the **Equations** window will look as follows:



$$\text{a-point}_i \sim \mathrm{N}(XB, \Omega)$$

$$\text{a-point}_i = \beta_{0i}\text{cons}$$

$$\beta_{0i} = 3.519(0.038) + e_{0i}$$

$$\left[e_{0i}\right] \sim \mathrm{N}(0, \Omega_e) : \Omega_e = \left[3.092(0.094)\right]$$

Deviance(MCMC) = 8589.826(2166 of 2166 cases in use)

Here we see that on average students get 3.519 (somewhere between a Grade C and D) in their A level chemistry exam but there is a large variability in scores (3.092). If we look at the DIC diagnostic (available via the **MCMC** submenu of the **Model** menu) we get the following value which we will use for comparison later:

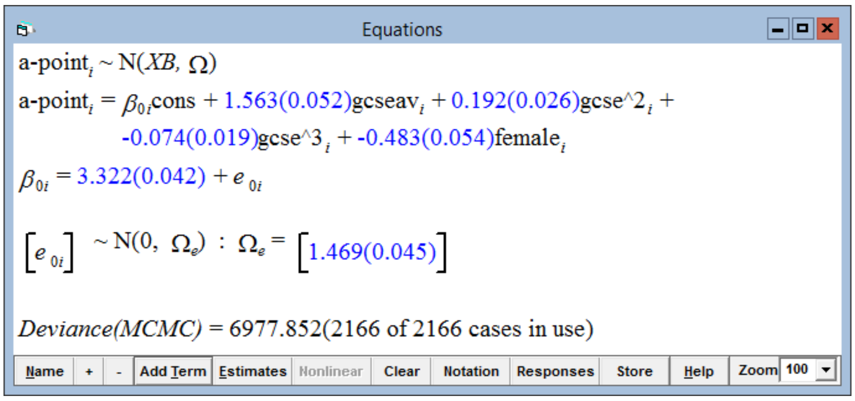
Dbar	D(thetabar)	pD	DIC
8589.83	8587.82	2.01	8591.84

We will now add in the three powers of the GCSE predictor and the gender predictor.

- Change **Estimation method** to **IGLS**.
- Select **Add Term** from the **Equations** window.

- Select **gcseav** from the **variable** list and click on the **Done** button.
- Repeat this procedure 3 times adding in **gcse^2**, **gcse^3** and **gender**.
- Click on the **Start** button.
- Change **Estimation method** to **MCMC**.
- Click on the **Start** button.

The **Equations** window will then look as follows:



$$\text{a-point}_i \sim \text{N}(XB, \Omega)$$
$$\text{a-point}_i = \beta_{0i}\text{cons} + 1.563(0.052)\text{gcseav}_i + 0.192(0.026)\text{gcse\^{}2}_i + -0.074(0.019)\text{gcse\^{}3}_i + -0.483(0.054)\text{female}_i$$
$$\beta_{0i} = 3.322(0.042) + e_{0i}$$
$$\left[e_{0i}\right] \sim \text{N}(0,\ \Omega_e)\ :\ \Omega_e = \left[1.469(0.045)\right]$$

Deviance(MCMC) = 6977.852(2166 of 2166 cases in use)

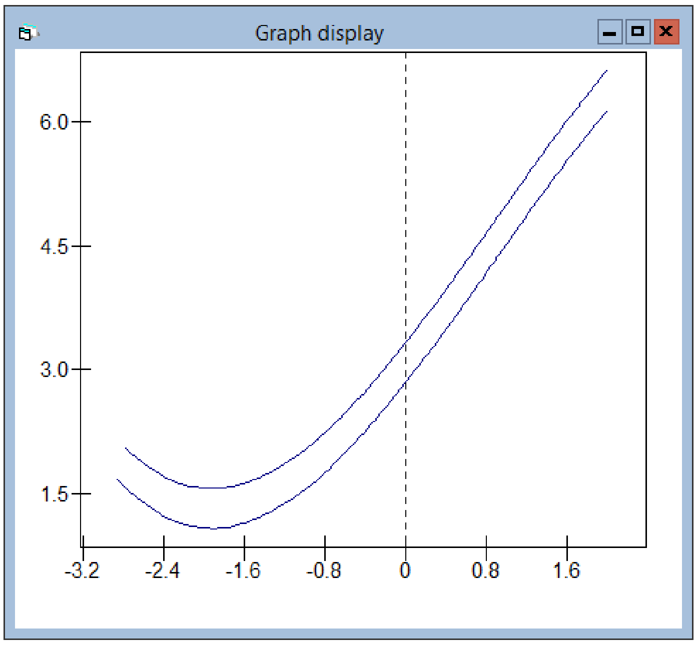
We firstly see that all these predictors are significant and that in combination they have reduced the unexplained variation from 3.092 to 1.469 that is over half the unknown variation has been explained. The DIC diagnostic as expected gives a much improved DIC value:

Dbar	D(thetabar)	pD	DIC	
6977.85	6971.86	5.99	6983.84	
8589.83	8587.82	2.01	8591.84	(intercept only model)

We can plot the relationship by calculating predictions via the **predictions** window and the **Customised graph** window as follows:

- Select **Predictions** from the **Model** menu.
- Click on the word **fixed** towards the bottom of the window and select **Include all fixed coefficients**.
- Select **c13** from the **output from prediction to** list.
- Click on the **Calc** button and close the window.
- Select **Customised Graph(s)** from the **Graphs** menu.
- Select **c13** as the y variable and **gcseav** as the x variable.
- Select **gender** as the group indicator and **line** as the plot type.
- Click on the **Apply** button.

The predictions graph will look as follows:



Here we have two parallel curves with the girls doing on average about half a grade worse than the boys (-0.483). The tail behaviour is rather unusual with the average predicted A level grade increasing once the average GCSE goes below 4 (-2 on the graph). This is due to four individuals who had low GCSE averages but achieved good A level chemistry grades (2 C's and 2 D's). If we however consider the range of GCSE averages greater than 4.5 (-1.5 on the graph and 95% of the pupils) we see a linear relationship. We will now look at treating the response variable as an ordered categorical response rather than as a continuous measure.

13.3 Ordered multinomial modelling

In the last chapter we considered unordered categorical models and saw how effectively such a model is a direct extension of the binary response model where we look at the relative probability of each category as compared to one base category and ensure that the probabilities of being in each category will sum to 1 for each individual. To exploit the ordering of the categories we do not model the probabilities of the individual categories but instead model the cumulative probabilities, for example the probability of getting a C grade or better, or as in our example the alternative probability of getting a D grade or worse.

To fit such a model we will therefore have to construct a set of indicator variables for each response. So if y_i is the response for individual i then we will define $y_i^{(s)}$ to be an indicator that takes value 1 if y_i is less than or equal to category s and 0 otherwise.

Then we will fit separate equations for each $y_i^{(s)}$ as follows:

$$E(y_i^{(s)}) = \gamma_i^{(s)} = \sum_{h=1}^{s} \pi_i^{(h)}, \qquad s = 1, \ldots, t-1$$

Here we assume there are t categories and that category t is the base category. In the example that follows we will assume that A is the base category and that we consider the less than a chosen category relationship but in practice we could have alternatively chosen F as the base category and used a greater than relationship.

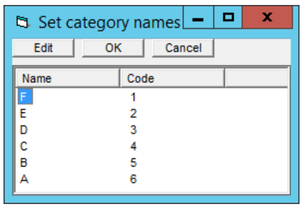
Now we need to define a link function to relate the probabilities $\gamma_i^{(s)}$ to the predictor variables. A common choice of model is the proportional odds model which corresponds to fitting a logit link. We can write this as

$$\text{logit}(\gamma_i^{(s)}) = \alpha^{(s)} + (X\beta)_i$$

Here we see that we allow an additive effect for each category but assume a common effect for all other predictor variables. This is one major difference between the ordered and unordered models but does make sense. In unordered models predictors might influence selecting different categories but in the ordered case the predictors can only have an effect of increasing/decreasing the chance of achieving a higher category.

We will now consider setting up a basic multinomial model. We firstly need to create the categorical response variable.

- Select **Names** from the **Data Manipulation** menu.
- Click on **a-point** in the list of names and then click on the **View** button in the categories section.
- Check that the **Categories** window is as shown below.
- Click on the **OK** button.



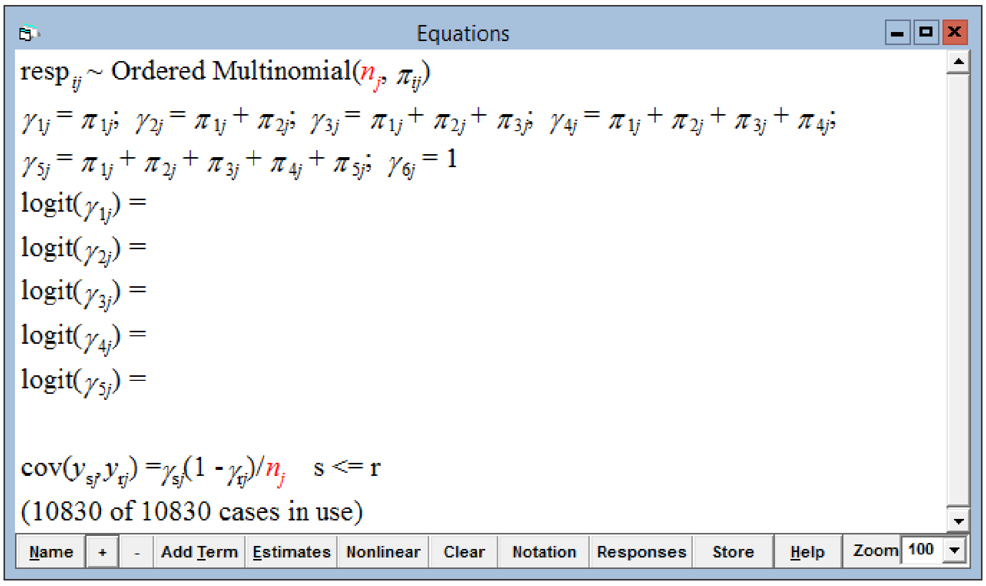
Name	Code
F	1
E	2
D	3
C	4
B	5
A	6

We will now set up the first categorical model in the **Equations** window.

- Select **Equations** from the **Model** menu.
- Change **Estimation mode** back to **IGLS**.
- Click on the **Clear** button to remove the Normal response model.

- Click on the red y and again select **a-point** as the y variable.
- Select **1-i** as the number of models and **pupil** as **level 1(i)** identifier.
- Click on the **Done** button.
- Click on the **N** and from the list of distributions that appears scroll down and select **multinomial**.
- In the **Multinomial options** box select **Ordered proportional odds** and **A** as the reference category.
- Click on the **Done** button.

The **Equations** window will now look as follows (after pressing the + button twice):


Equations

$\text{resp}_{ij} \sim \text{Ordered Multinomial}(n_j, \pi_{ij})$

$\gamma_{1j} = \pi_{1j};\ \gamma_{2j} = \pi_{1j} + \pi_{2j};\ \gamma_{3j} = \pi_{1j} + \pi_{2j} + \pi_{3j};\ \gamma_{4j} = \pi_{1j} + \pi_{2j} + \pi_{3j} + \pi_{4j};$

$\gamma_{5j} = \pi_{1j} + \pi_{2j} + \pi_{3j} + \pi_{4j} + \pi_{5j};\ \gamma_{6j} = 1$

$\text{logit}(\gamma_{1j}) =$

$\text{logit}(\gamma_{2j}) =$

$\text{logit}(\gamma_{3j}) =$

$\text{logit}(\gamma_{4j}) =$

$\text{logit}(\gamma_{5j}) =$

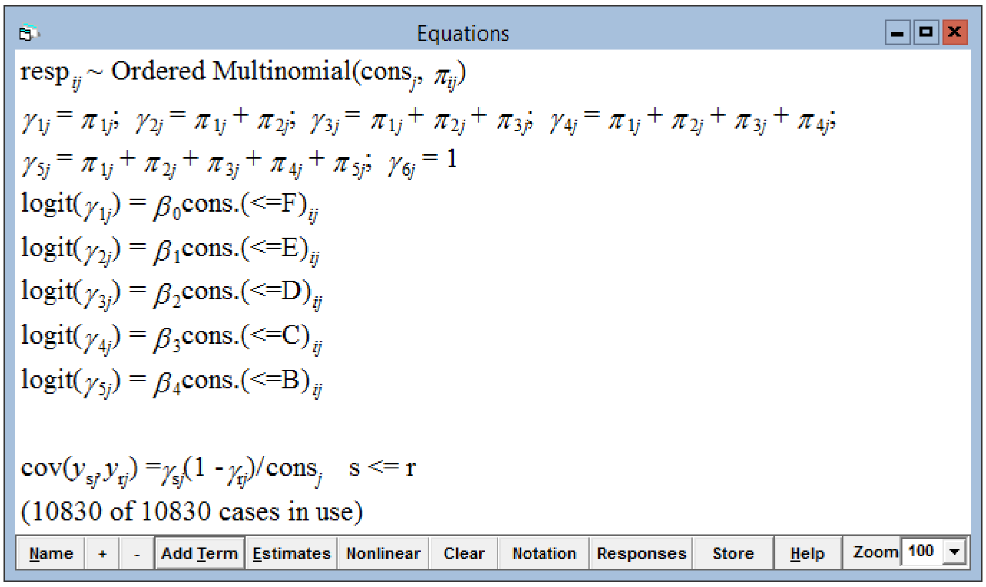
$\text{cov}(y_{sj}, y_{rj}) = \gamma_{sj}(1 - \gamma_{rj})/n_j \quad s <= r$

(10830 of 10830 cases in use)

You will notice in the above window that the model now has two levels and a new response has been created in column **resp**. We can finish setting up the first model as follows:

- Click on the n_j in the window and select **cons** as the denominator.
- Click on the **Done** button.
- Click on the **Add Term** button and select **cons** from the variable list.
- Click on the **add Separate coefficients** button.

The **Equations** window will now look as follows:



$\text{resp}_{ij} \sim \text{Ordered Multinomial}(\text{cons}_j, \pi_{ij})$

$\gamma_{1j} = \pi_{1j};\ \gamma_{2j} = \pi_{1j} + \pi_{2j};\ \gamma_{3j} = \pi_{1j} + \pi_{2j} + \pi_{3j};\ \gamma_{4j} = \pi_{1j} + \pi_{2j} + \pi_{3j} + \pi_{4j};$

$\gamma_{5j} = \pi_{1j} + \pi_{2j} + \pi_{3j} + \pi_{4j} + \pi_{5j};\ \gamma_{6j} = 1$

$\text{logit}(\gamma_{1j}) = \beta_0 \text{cons.}(<=F)_{ij}$

$\text{logit}(\gamma_{2j}) = \beta_1 \text{cons.}(<=E)_{ij}$

$\text{logit}(\gamma_{3j}) = \beta_2 \text{cons.}(<=D)_{ij}$

$\text{logit}(\gamma_{4j}) = \beta_3 \text{cons.}(<=C)_{ij}$

$\text{logit}(\gamma_{5j}) = \beta_4 \text{cons.}(<=B)_{ij}$

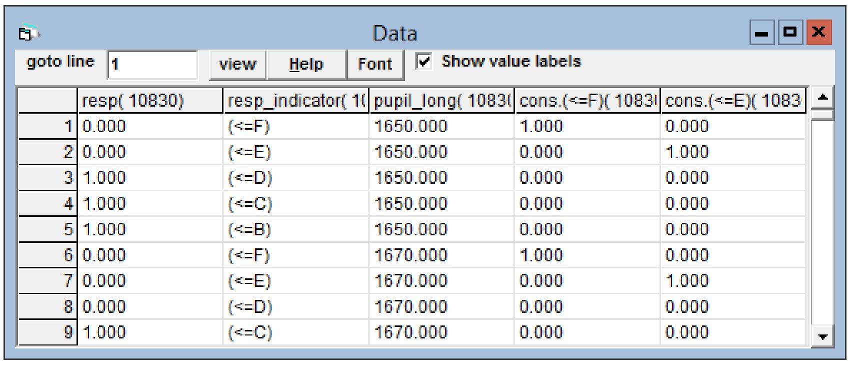
$\text{cov}(y_{sj}, y_{rj}) = \gamma_{sj}(1 - \gamma_{rj})/\text{cons}_j \quad s <= r$

(10830 of 10830 cases in use)

We can examine the new columns created via the **Data** window.

- Select **View or Edit Data** from the **Data Manipulation** menu.
- Click on the **view** button.
- Select the columns **resp**, **resp_indicator**, **pupil_long**, **cons.(<=F)** and **cons.(<=E)** using the CTRL keyto select multiple columns.
- Click on the **OK** button.

The **Data** window will then appear as follows. Here we can see that the original response for each individual has been converted into 5 indicator variables stacked in the column **resp**. Here for example the first pupil (pupil 1650) achieved a D and this is converted into the pattern of 5 indicators (0, 0, 1, 1, 1). The **resp_indicator** column gives information on what each indicator indicates and is used as the level 1 identifier. The original pupil column has been replicated to produce the level 2 indicators, **pupil_long**. The 5 indicators each have an associated intercept, which are picked up by the 5 new constant columns.

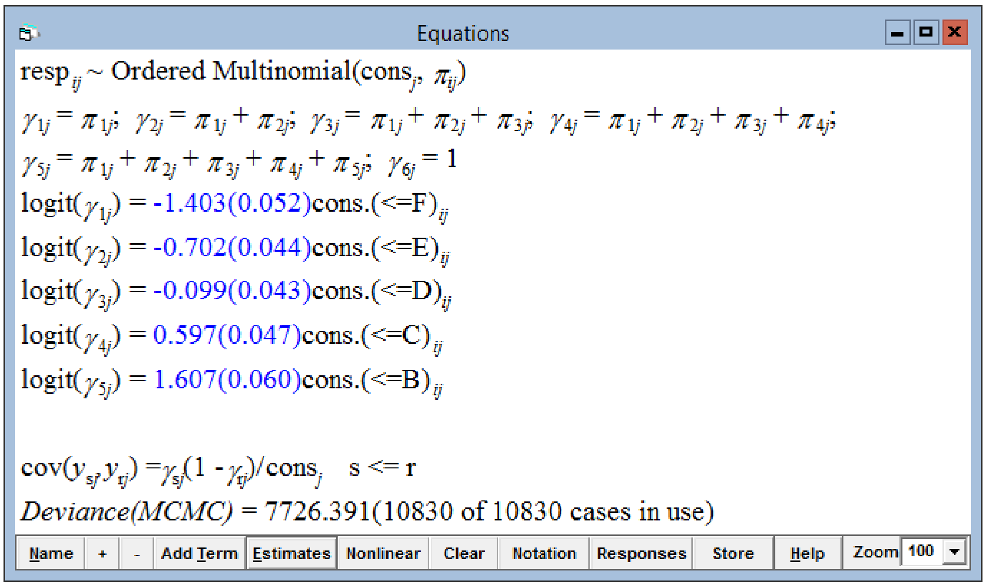


	resp(10830)	resp_indicator(1(	pupil_long(1083(	cons.(<=F)(1083(	cons.(<=E)(1083
1	0.000	(<=F)	1650.000	1.000	0.000
2	0.000	(<=E)	1650.000	0.000	1.000
3	1.000	(<=D)	1650.000	0.000	0.000
4	1.000	(<=C)	1650.000	0.000	0.000
5	1.000	(<=B)	1650.000	0.000	0.000
6	0.000	(<=F)	1670.000	1.000	0.000
7	0.000	(<=E)	1670.000	0.000	1.000
8	0.000	(<=D)	1670.000	0.000	0.000
9	1.000	(<=C)	1670.000	0.000	0.000

We will now run the model using MCMC estimation.

- Select **Equations** from the **Model** menu.
- Click on the **Nonlinear** button.
- On the screen that appears click on the **Use Defaults** and **Done** buttons.
- Click on the **Start** button.
- Change **Estimation method** to **MCMC**.
- Click on the **Start** button.

Upon finishing the 5,000 iterations the **Equations** window will appear as follows (after pressing the Estimates button twice):



$\text{resp}_{ij} \sim \text{Ordered Multinomial}(\text{cons}_j, \pi_{ij})$

$\gamma_{1j} = \pi_{1j};\ \gamma_{2j} = \pi_{1j} + \pi_{2j};\ \gamma_{3j} = \pi_{1j} + \pi_{2j} + \pi_{3j};\ \gamma_{4j} = \pi_{1j} + \pi_{2j} + \pi_{3j} + \pi_{4j};$

$\gamma_{5j} = \pi_{1j} + \pi_{2j} + \pi_{3j} + \pi_{4j} + \pi_{5j};\ \gamma_{6j} = 1$

$\text{logit}(\gamma_{1j}) = -1.403(0.052)\text{cons.}(<=F)_{ij}$

$\text{logit}(\gamma_{2j}) = -0.702(0.044)\text{cons.}(<=E)_{ij}$

$\text{logit}(\gamma_{3j}) = -0.099(0.043)\text{cons.}(<=D)_{ij}$

$\text{logit}(\gamma_{4j}) = 0.597(0.047)\text{cons.}(<=C)_{ij}$

$\text{logit}(\gamma_{5j}) = 1.607(0.060)\text{cons.}(<=B)_{ij}$

$\text{cov}(y_{sj}, y_{rj}) = \gamma_{sj}(1 - \gamma_{rj})/\text{cons}_j \quad s <= r$

Deviance(MCMC) = 7726.391(10830 of 10830 cases in use)

We can use these coefficients to calculate the probabilities of achieving each of the 6 grades. For example the estimated probability of getting a grade F is the anti-logit of -1.403, which equals 0.197. Similarly the probability of getting a grade E is the anti-logit of -0.703 minus the probability of getting an F, i.e. $0.331 - 0.197 = 0.134$. For this simple model we could of course have calculated these probabilities by simply working out empirically the proportion of pupils in the dataset who got each grade. The modelling approach comes into its own when we add predictors. However firstly we will calculate the DIC diagnostic for this first model for comparison later. Selecting **MCMC/DIC diagnostic** from the **Model** menu gives the following:

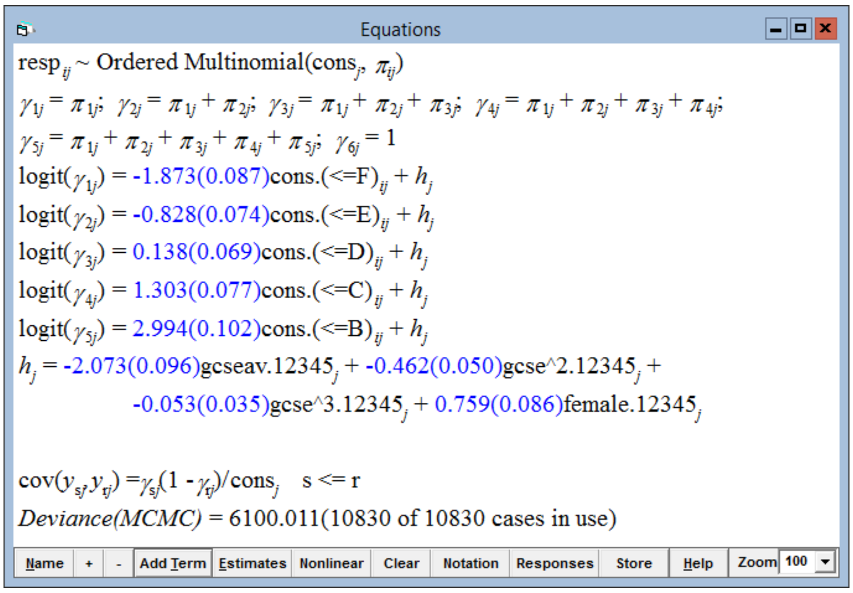
Dbar	D(thetabar)	pD	DIC
7726.39	7721.42	4.97	7731.36

13.4 Adding predictor variables

In the Normal response modelling section our next step was to include predictors into the model. To do this we do the following:

- Change **Estimation method** back to **IGLS**.
- Select **Equations** from the **Model** menu.
- Click on the **Add Term** button.
- Select **gcseav** from the variable list.
- Click on the **add Common coefficient** button.
- Click on the **Include all** and **Done** buttons.
- Repeat this procedure for the variables **gcse^2**, **gcse^3** and **gender**.
- Click on the **Start** button
- Change **Estimation method** to **MCMC**.
- Click on the **Start** button.

After 5,000 iterations we get the following results:



$\text{resp}_{ij} \sim \text{Ordered Multinomial}(\text{cons}_j, \pi_{ij})$

$\gamma_{1j} = \pi_{1j};\ \gamma_{2j} = \pi_{1j} + \pi_{2j};\ \gamma_{3j} = \pi_{1j} + \pi_{2j} + \pi_{3j};\ \gamma_{4j} = \pi_{1j} + \pi_{2j} + \pi_{3j} + \pi_{4j};$

$\gamma_{5j} = \pi_{1j} + \pi_{2j} + \pi_{3j} + \pi_{4j} + \pi_{5j};\ \gamma_{6j} = 1$

$\text{logit}(\gamma_{1j}) = -1.873(0.087)\text{cons.}(<=F)_{ij} + h_j$

$\text{logit}(\gamma_{2j}) = -0.828(0.074)\text{cons.}(<=E)_{ij} + h_j$

$\text{logit}(\gamma_{3j}) = 0.138(0.069)\text{cons.}(<=D)_{ij} + h_j$

$\text{logit}(\gamma_{4j}) = 1.303(0.077)\text{cons.}(<=C)_{ij} + h_j$

$\text{logit}(\gamma_{5j}) = 2.994(0.102)\text{cons.}(<=B)_{ij} + h_j$

$h_j = -2.073(0.096)\text{gcseav.12345}_j + -0.462(0.050)\text{gcse}\hat{}2.12345_j +$
$-0.053(0.035)\text{gcse}\hat{}3.12345_j + 0.759(0.086)\text{female.12345}_j$

$\text{cov}(y_{sj}, y_{rj}) = \gamma_{sj}(1 - \gamma_{rj})/\text{cons}_j \quad s <= r$

Deviance(MCMC) = 6100.011(10830 of 10830 cases in use)

Here we see that all the predictors have negative coefficients. This is because we are predicting the probability of getting less than or equal to particular grades and so as the ability of the pupil increases they are less likely to get below a particular threshold. The coefficients suggest that we do not need to fit a cubic term here and if we remove it and compare the DIC diagnostic we see the following:

Dbar	D(thetabar)	pD	DIC	
6100.01	6090.94	9.07	6109.09	(with cubic term)
6100.97	6092.71	8.26	6109.23	(without cubic term)

It should be noted, as shown in the User's Guide to MLwiN, that we could in fact fit separate effects of each predictor for each grade boundary but as shown there fitting a single common predictor effect is acceptable. We will now consider fitting a multilevel multinomial model, where the probabilities of getting the different grades depend on the school the students belong to.

13.5 Multilevel ordered response modelling

As with all the other response types that we have considered in this book it is perfectly plausible for there to be random variation between higher level units and ordered response models are no exception.

We will need to define the column that contains the higher level units but firstly we need to remove the cubic term from the model (if we have not already done so)

- Change **Estimation method** to **IGLS**.
- Select **Equations** from the **Model** menu.
- Click on the term **gcse^3.12345** and click on the **Delete Term** button.
- Click on the term **resp** to bring up the **Y variable** window.
- Change the number of levels indicator to **3-ijk**.
- Select **estab** as the **level 3(k)** identifier and click on the **Done** button.

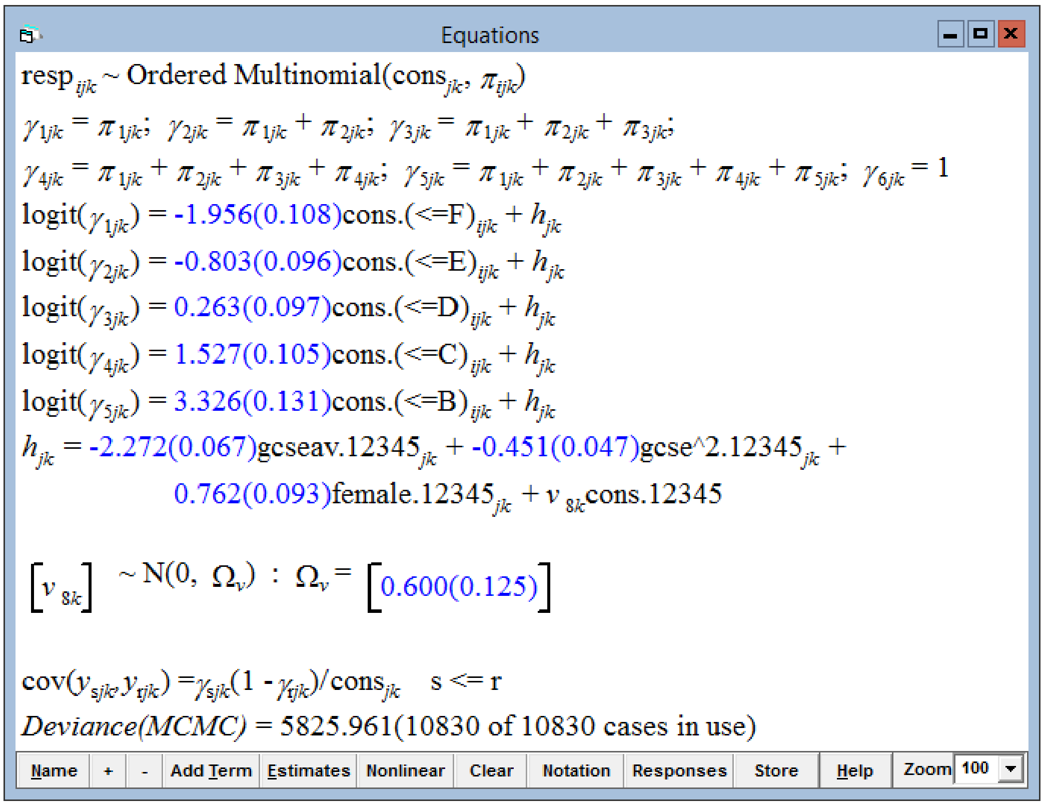
We now need to decide how to add in the effect of the different schools. If we were to add random effects for each grade boundary then we would effectively be ignoring the ordering and would have an equivalent model to the unordered responses. We therefore wish to have one effect for each school that changes each grade boundary. To obtain this we do the following:

- Click on the **Add Term** button.
- Select **cons** from the variable list.
- Click on the **add Common coefficient** button.
- Click on the **Include all** and **Done** buttons.

This will have created a variable **cons.12345** which is a common intercept term. We do not want to fit this as a fixed effect but instead want to fit it as random at the establishment (school) level. To do this we do the following:

- Click on the **cons.12345** term.
- In the **X variable** window that appears tick the **k(estab_long)** tickbox.
- Remove the tick in the **Fixed Parameter** tickbox.
- Click on the **Done** button.
- Click on the **Start** button.
- Change **Estimation method** to **MCMC**.
- Click on the **Start** button.

Upon completion of the iterations we get the following estimates:



$\text{resp}_{ijk} \sim \text{Ordered Multinomial}(\text{cons}_{jk}, \pi_{ijk})$

$\gamma_{1jk} = \pi_{1jk};\ \gamma_{2jk} = \pi_{1jk} + \pi_{2jk};\ \gamma_{3jk} = \pi_{1jk} + \pi_{2jk} + \pi_{3jk};$

$\gamma_{4jk} = \pi_{1jk} + \pi_{2jk} + \pi_{3jk} + \pi_{4jk};\ \gamma_{5jk} = \pi_{1jk} + \pi_{2jk} + \pi_{3jk} + \pi_{4jk} + \pi_{5jk};\ \gamma_{6jk} = 1$

$\text{logit}(\gamma_{1jk}) = -1.956(0.108)\text{cons.(<=F)}_{ijk} + h_{jk}$

$\text{logit}(\gamma_{2jk}) = -0.803(0.096)\text{cons.(<=E)}_{ijk} + h_{jk}$

$\text{logit}(\gamma_{3jk}) = 0.263(0.097)\text{cons.(<=D)}_{ijk} + h_{jk}$

$\text{logit}(\gamma_{4jk}) = 1.527(0.105)\text{cons.(<=C)}_{ijk} + h_{jk}$

$\text{logit}(\gamma_{5jk}) = 3.326(0.131)\text{cons.(<=B)}_{ijk} + h_{jk}$

$h_{jk} = -2.272(0.067)\text{gcseav.12345}_{jk} + -0.451(0.047)\text{gcse\^{}2.12345}_{jk} + 0.762(0.093)\text{female.12345}_{jk} + v_{8k}\text{cons.12345}$

$\left[v_{8k}\right] \sim \text{N}(0,\ \Omega_v) \ :\ \Omega_v = \left[0.600(0.125)\right]$

$\text{cov}(y_{sjk}, y_{rjk}) = \gamma_{sjk}(1 - \gamma_{rjk})/\text{cons}_{jk} \quad s <= r$

Deviance(MCMC) = 5825.961(10830 of 10830 cases in use)

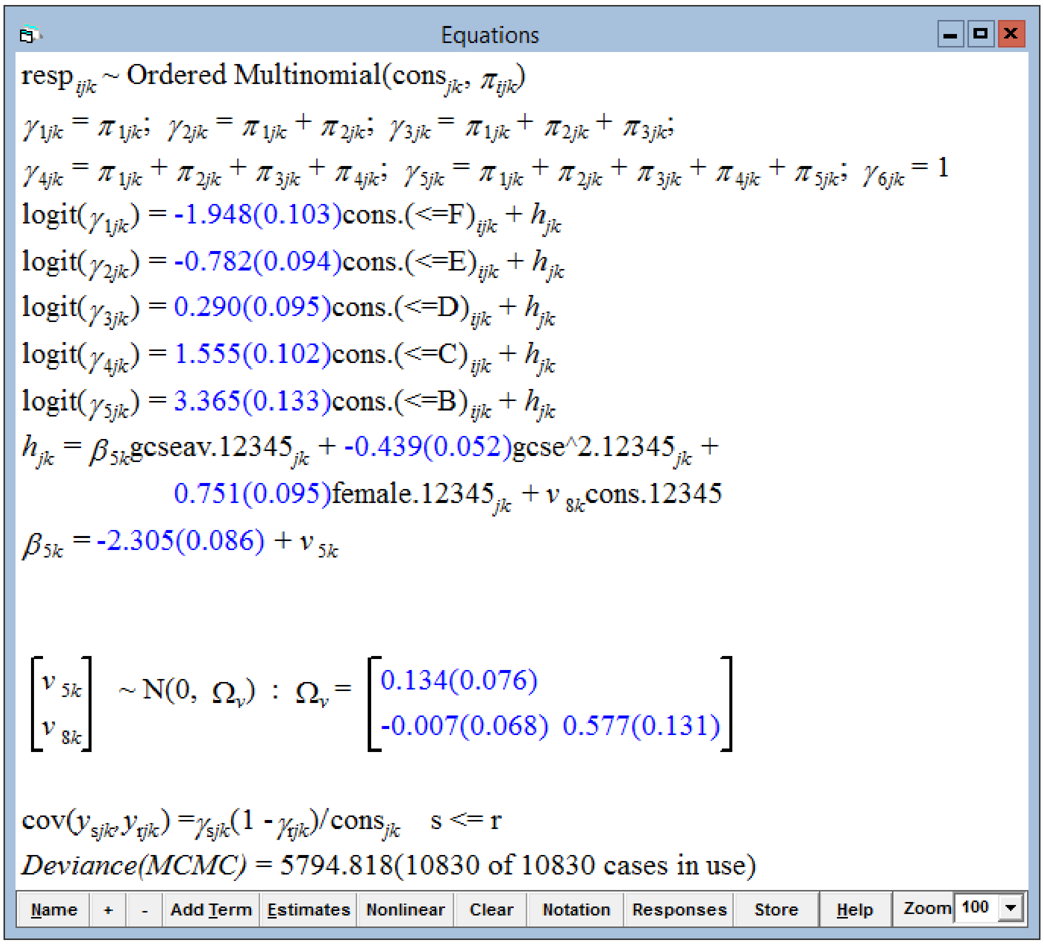
Here we see that there is some variability between the various establishments and fitting this model accounts for this variability. Looking at the DIC diagnostic we see that adding the 219 random effects has given rise to 110 additional 'effective' parameters. The DIC has also reduced by over 150 suggesting a much better fit to the data.

Dbar	D(thetabar)	pD	DIC	
5825.96	5707.91	118.05	5944.01	(with random effects)
6100.97	6092.71	8.26	6109.23	(without random effects)

We can continue to fit more complex models for example we may think that the effect of GCSE score is different for each school, which would result in a random slopes model. To fit such a model we do the following:

- Change **Estimation method** to **IGLS**.
- Click on the term **gcseav.12345** in the **Equations** window.
- In the **X variable** window that appears tick the **k(estab_long)** tickbox.
- Click on the **Done** button.
- Click on the **Start** button.
- Change **Estimation method** to **MCMC**.
- Click on the **Start** button.

After running for 5,000 iterations we get the following estimates:



$\text{resp}_{ijk} \sim \text{Ordered Multinomial}(\text{cons}_{jk}, \pi_{ijk})$

$\gamma_{1jk} = \pi_{1jk};\ \gamma_{2jk} = \pi_{1jk} + \pi_{2jk};\ \gamma_{3jk} = \pi_{1jk} + \pi_{2jk} + \pi_{3jk};$

$\gamma_{4jk} = \pi_{1jk} + \pi_{2jk} + \pi_{3jk} + \pi_{4jk};\ \gamma_{5jk} = \pi_{1jk} + \pi_{2jk} + \pi_{3jk} + \pi_{4jk} + \pi_{5jk};\ \gamma_{6jk} = 1$

$\text{logit}(\gamma_{1jk}) = -1.948(0.103)\text{cons.}(<=\text{F})_{ijk} + h_{jk}$

$\text{logit}(\gamma_{2jk}) = -0.782(0.094)\text{cons.}(<=\text{E})_{ijk} + h_{jk}$

$\text{logit}(\gamma_{3jk}) = 0.290(0.095)\text{cons.}(<=\text{D})_{ijk} + h_{jk}$

$\text{logit}(\gamma_{4jk}) = 1.555(0.102)\text{cons.}(<=\text{C})_{ijk} + h_{jk}$

$\text{logit}(\gamma_{5jk}) = 3.365(0.133)\text{cons.}(<=\text{B})_{ijk} + h_{jk}$

$h_{jk} = \beta_{5k}\text{gcseav.12345}_{jk} + -0.439(0.052)\text{gcse\^{}2.12345}_{jk} +$
$\quad 0.751(0.095)\text{female.12345}_{jk} + v_{8k}\text{cons.12345}$

$\beta_{5k} = -2.305(0.086) + v_{5k}$

$$\begin{bmatrix} v_{5k} \\ v_{8k} \end{bmatrix} \sim \text{N}(0,\ \Omega_v)\ :\ \Omega_v = \begin{bmatrix} 0.134(0.076) & \\ -0.007(0.068) & 0.577(0.131) \end{bmatrix}$$

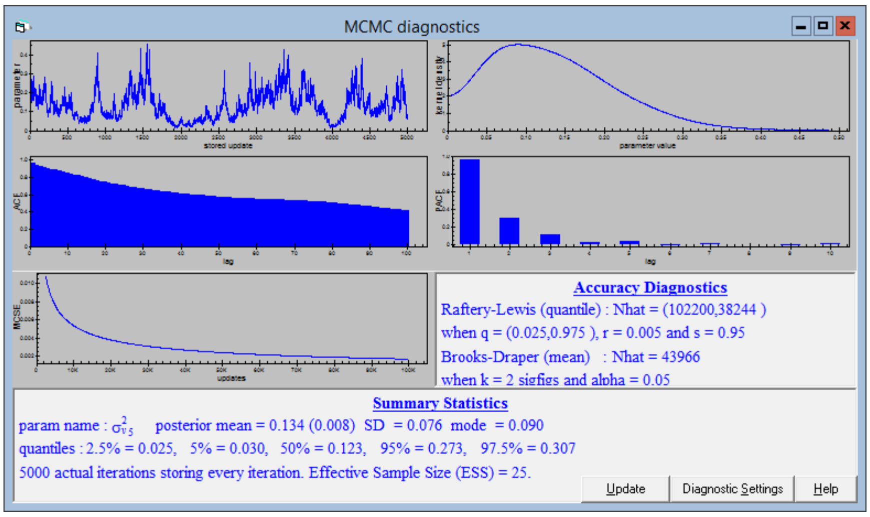
$\text{cov}(y_{sjk}, y_{rjk}) = \gamma_{sjk}(1 - \gamma_{rjk})/\text{cons}_{jk} \quad s <= r$

Deviance(MCMC) = 5794.818(10830 of 10830 cases in use)

If we compare this model with the model that does not include random effects for GCSE score via the DIC diagnostic we get the following:

Dbar	D(thetabar)	pD	DIC	
5794.82	5649.08	145.74	5940.56	(with random slopes)
5825.69	5707.91	118.05	5944.01	(without random slopes)

So the addition of random GCSE effects only makes a marginal improvement. It should be noted that throughout this chapter we have only been running for 5,000 monitoring iterations for each model. If we look at the chain for the variance of the random GCSE effects via the **Trajectories** window we will see the following:



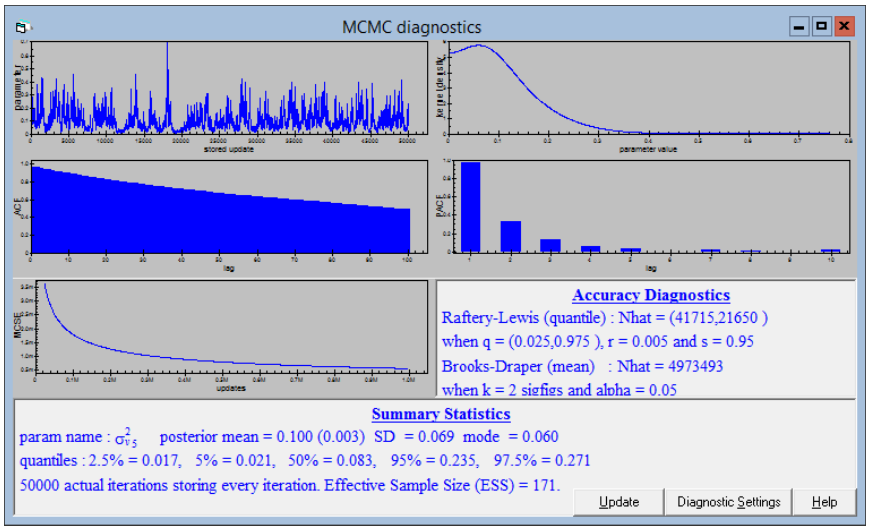
Accuracy Diagnostics
Raftery-Lewis (quantile) : Nhat = (102200,38244)
when q = (0.025,0.975), r = 0.005 and s = 0.95
Brooks-Draper (mean) : Nhat = 43966
when k = 2 sigfigs and alpha = 0.05

Summary Statistics
param name : σ^2_{v5} posterior mean = 0.134 (0.008) SD = 0.076 mode = 0.090
quantiles : 2.5% = 0.025, 5% = 0.030, 50% = 0.123, 95% = 0.273, 97.5% = 0.307
5000 actual iterations storing every iteration. Effective Sample Size (ESS) = 25.

Here we see that clearly more iterations are required. We will therefore run for 50,000 iterations instead.

- Click on the **Estimation Control** button.
- Change the **Monitoring chain length** to 50000.
- Press the **More** button.

The MCMC diagnostics after 50,000 iterations (which will take several minutes) are as follows:



Accuracy Diagnostics

Raftery-Lewis (quantile) : Nhat = (41715,21650)
when q = (0.025,0.975), r = 0.005 and s = 0.95
Brooks-Draper (mean) : Nhat = 4973493
when k = 2 sigfigs and alpha = 0.05

Summary Statistics

param name : σ^2_{v5} posterior mean = 0.100 (0.003) SD = 0.069 mode = 0.060
quantiles : 2.5% = 0.017, 5% = 0.021, 50% = 0.083, 95% = 0.235, 97.5% = 0.271
50000 actual iterations storing every iteration. Effective Sample Size (ESS) = 171.

Here we see that we have now run for sufficiently long enough to satisfy the diagnostics and in fact the parameter estimate has changed very little. If we look at the DIC diagnostic we see the following:

Dbar	D(thetabar)	pD	DIC	
5799.36	5658.88	140.48	5939.83	(after 50,000 iterations)
5794.82	5649.08	145.74	5940.56	(after 5,000 iterations)

Here the additional run length has confirmed a reasonably similar value for the DIC diagnostic.

We could now extend our modelling as in the User's Guide to MLwiN and consider the effect of making gender random at the establishment level but we leave this to the reader. As described in the last chapter an advantage of using a simulation based technique like MCMC is that we can create chains for derived parameters and this can also be done in ordered response models to create interval estimates for conditional probabilities of achieving particular grades. Again we leave this as an exercise for the interested reader.

Chapter learning outcomes

- ⋆ How to fit models to ordered categorical responses
- ⋆ How to extend such models to include random effects
- ⋆ The differences between ordered and unordered categorical response models

Chapter 14

Adjusting for Measurement Errors in Predictor Variables

Statistical modelling attempts to fit models that describe the relationship between an observed response variable and several observed predictor variables. We have so far considered how to fit models where the response variable is continuous, a binary indicator or a count. All these models assume that there is variation between the observed data and a predicted model, and that this variation is due to additional unobserved predictor variables. When choosing statistical models we try to find a balance between the fit and complexity of a model. If we were to continue adding in predictors we could achieve a perfect fit to the dataset, but adding in 'non-significant' predictors will reduce the predictive power of our model for observations outside of the dataset.

Apart from variation due to unknown predictors, there may also be variation or errors in the predictors that we fit in our model. For example if we consider the tutorial dataset that was first introduced in Chapter 1 and focus on the London Reading Test (LRT) predictor that was used as a proxy for intake ability, then errors could occur in this predictor in many ways. Firstly obvious measurement errors could occur, for example the teacher could mark a paper wrongly or the researcher who created the dataset could type the data incorrectly into the computer. Secondly, less obvious errors could occur if we consider the variable as a proxy for intake ability, for example a child may do better on the test, as his favourite types of questions occur, while another child may be ill on the day of the test and hence do worse.

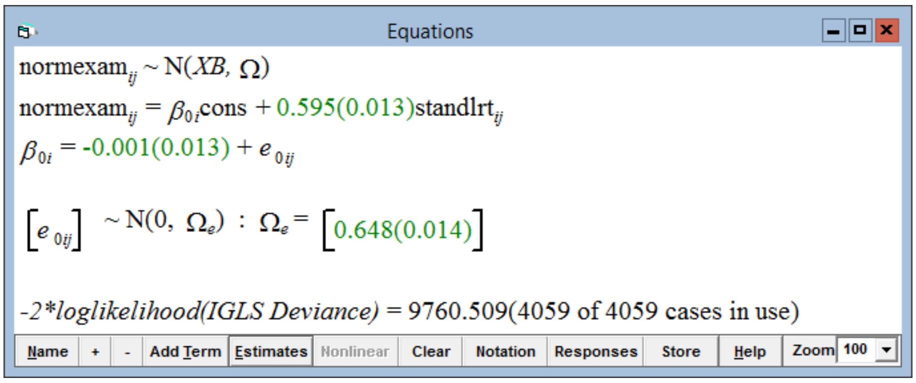
Measurement errors can occur in many forms. If the predictor is (pseudo) continuous like LRT score then we can assume that the measurement error is also continuous and these types of measurement error can currently be fitted in MLwiN. If however the predictor was a categorical variable for example an exam grade or the sex of the pupil then errors in these variables can be thought of as misclassifications, for example a boy is misclassified as a girl, and currently methods for dealing with these types of error have not been implemented in MLwiN.

14.1 Effects of measurement error on predictors

To illustrate the effect of measurement error on a predictor we will look at fitting a simple linear regression model to the **tutorial** dataset and then introducing errors into the LRT predictor.

- Select **Open Sample Worksheet** from the **File** menu.
- Select **tutorial.ws** from the list and click on **Open**.
- Set up the OLS linear regression model in the **Equations** window as in Chapter 2.

You should now have set **normexam** as the response, **school** as the level 2 identifier, **student** as the level 1 identifier, **cons** as a fixed effect and random at level 1 and finally **standlrt** as a fixed effect (see Chapter 1 if you need more details). For speed we will here use the IGLS method so run the model by pressing the **Start** button. Upon convergence the model will look as follows:



$\text{normexam}_{ij} \sim \text{N}(XB, \Omega)$

$\text{normexam}_{ij} = \beta_{0i}\text{cons} + 0.595(0.013)\text{standlrt}_{ij}$

$\beta_{0i} = -0.001(0.013) + e_{0ij}$

$\begin{bmatrix} e_{0ij} \end{bmatrix} \sim \text{N}(0,\ \Omega_e)\ :\ \Omega_e = \begin{bmatrix} 0.648(0.014) \end{bmatrix}$

$-2*loglikelihood(IGLS\ Deviance) = 9760.509(4059\ \text{of}\ 4059\ \text{cases in use})$

Here (as **normexam** has variance 1) the predictor **standlrt** has explained $100 - 64.8 = 35.2\%$ of the variation in the response variable **normexam**. We will assume that the **standlrt** we have fitted is the 'true' predictor and now we will generate some random (Normal) errors. This we can do via the **Generate random number** window although below we will use the **Command interface** window. Firstly however we wish to look at how much variation there is in the 'true' predictor.

- Select **Averages and Correlation** from the **Basic Statistics** menu.

- Select **standlrt** from the drop down list and click on **Calculate**.

This will give the following output:

	N	Missing	Mean	s.d.
standlrt	4059	0	0.0018103	0.99322

So here we see that the (standardized) variable has a variance of approximately 1. We will now consider adding a measurement error to this variable that has variance 0.2. Note that in the measurement error literature the term reliability is often used with respect to measurement errors (see Woodhouse et al., 1996) and here we will have a reliability of $(1/(1+0.2))\times 100\%$ = 83.3%.

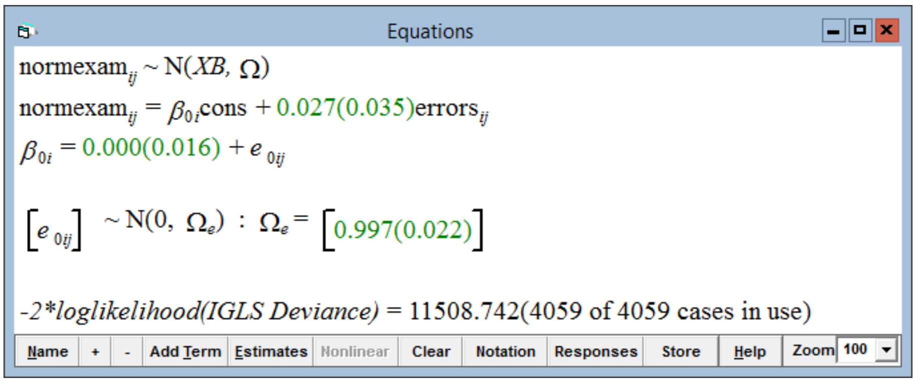
To generate the errors we will type the following commands in the **Command interface** window:

```
▶ Seed 1
▶ Nran 4059 c11
▶ Calc c11=c11*sqrt(0.2)
▶ Name c11 'errors'
▶ Calc c12 = 'errors' + 'standlrt'
▶ Name c12 'obslrt'
```

Here we set the random number seed to 1 so that you get the same set of simulated errors as the book. We then store the errors in column **c11** and the resulting observed predictor values in column **c12**. It will be firstly of interest to fit a linear regression of the errors against the **normexam** variable.

- In the **Equations** window click on β_1 (**standlrt**).
- Replace **standlrt** with **errors** from the pull down list.
- Click on the **Done** button.
- Click on the **Start** button.

Upon convergence the estimates will be as follows:


Equations
Name + - Add Term Estimates Nonlinear Clear Notation Responses Store Help Zoom 100

$\text{normexam}_{ij} \sim \text{N}(XB, \Omega)$

$\text{normexam}_{ij} = \beta_{0i}\text{cons} + 0.027(0.035)\text{errors}_{ij}$

$\beta_{0i} = 0.000(0.016) + e_{0ij}$

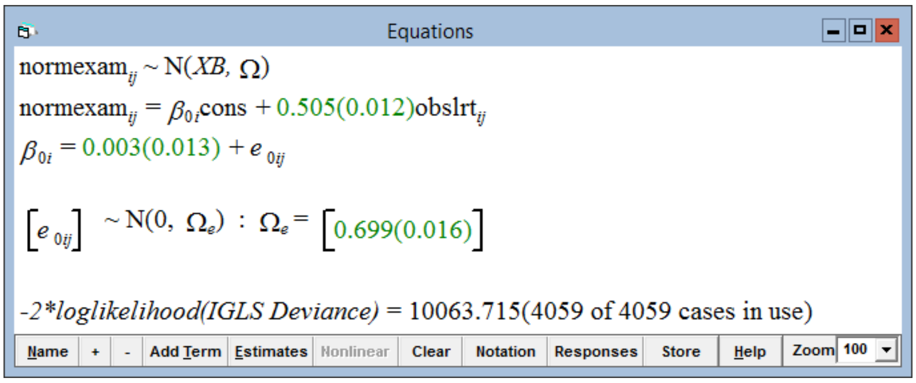
$\begin{bmatrix} e_{0ij} \end{bmatrix} \sim \text{N}(0, \Omega_e) : \Omega_e = \begin{bmatrix} 0.997(0.022) \end{bmatrix}$

*-2*loglikelihood(IGLS Deviance)* = 11508.742(4059 of 4059 cases in use)

So here we see a non-significant positive relationship between the errors and the exam score, which explains 0.3% of the variation. Of course the errors are random and so a different set of random errors may produce a negative relationship. We now look at the relationship between **normexam** and the observed (i.e. with errors) LRT scores:

- In the **Equations** window click on β_1 (**errors**).
- Replace **errors** with **obslrt** from the pull down list.
- Click on the **Done** button.
- Click on the **Start** button.

Upon convergence we will get the following:


Equations
Name + - Add Term Estimates Nonlinear Clear Notation Responses Store Help Zoom 100

$\text{normexam}_{ij} \sim \text{N}(XB, \Omega)$

$\text{normexam}_{ij} = \beta_{0i}\text{cons} + 0.505(0.012)\text{obslrt}_{ij}$

$\beta_{0i} = 0.003(0.013) + e_{0ij}$

$\begin{bmatrix} e_{0ij} \end{bmatrix} \sim \text{N}(0, \Omega_e) : \Omega_e = \begin{bmatrix} 0.699(0.016) \end{bmatrix}$

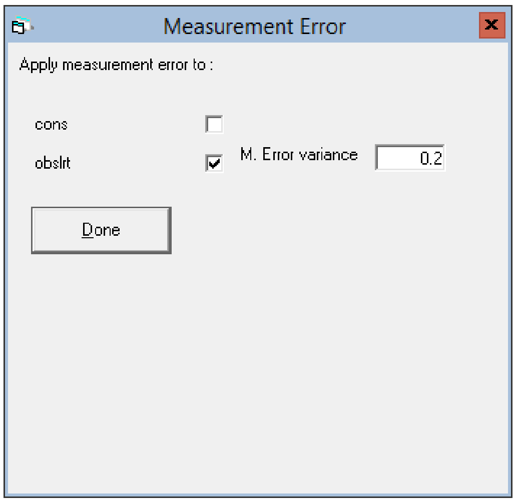
*-2*loglikelihood(IGLS Deviance)* = 10063.715(4059 of 4059 cases in use)

So in adding the measurement errors to the predictor we have reduced its predictive power and hence its coefficient is smaller and the residual variance has increased. We could think of this regression as a weighted average of the earlier regressions between the exam score with true LRT score and the exam score with errors. Here the weights will be approximately equal to the precisions (1/variance) of the estimates, although due to the intercepts being estimated and not fixed at zero this will not hold exactly.

In order now to adjust for the measurement errors we need to add some extra statements to the model. The MCMC methods used in MLwiN to adjust for measurement error involve extra steps in which we generate the true values at each iteration. To set up the MCMC measurement error model we do the following.

- Click on the **Estimation Control** button.
- Click on the **MCMC** tab.
- Select **MCMC/Measurement Errors** from the **Model** menu.
- Click on the tick box for **obslrt** on the **Measurement errors** window and input **0.2** as the error variance.

The window should then look as follows:

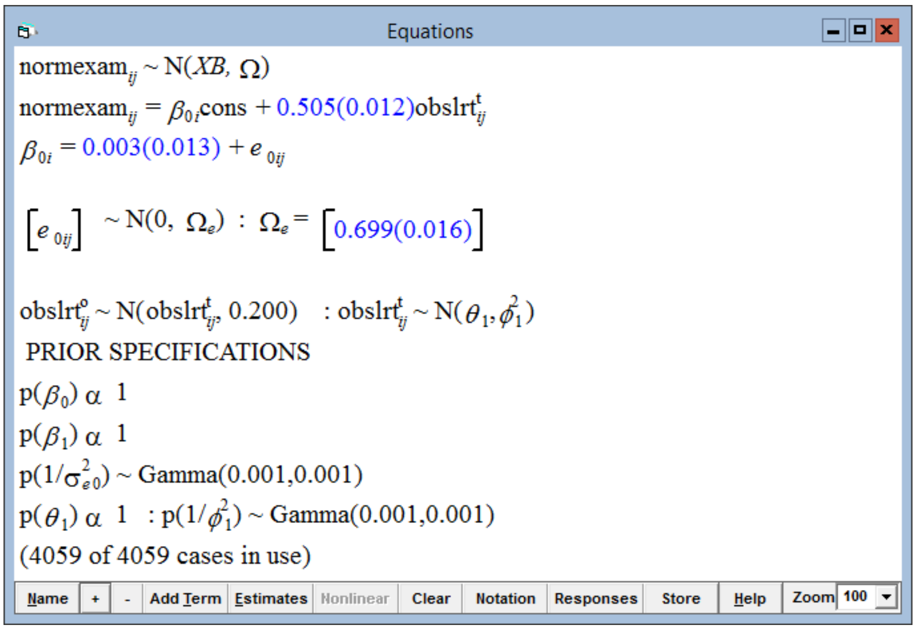


Apply measurement error to :

cons ☐

obslrt ☑ M. Error variance 0.2

If we now look at the **Equations** window (and press + to get the prior information) we will get a window as follows:



$\text{normexam}_{ij} \sim \text{N}(XB, \Omega)$

$\text{normexam}_{ij} = \beta_{0i}\text{cons} + 0.505(0.012)\text{obslrt}^{t}_{ij}$

$\beta_{0i} = 0.003(0.013) + e_{0ij}$

$\left[e_{0ij}\right] \sim \text{N}(0,\ \Omega_e)\ :\ \Omega_e = \left[0.699(0.016)\right]$

$\text{obslrt}^{o}_{ij} \sim \text{N}(\text{obslrt}^{t}_{ij}, 0.200)\quad : \text{obslrt}^{t}_{ij} \sim \text{N}(\theta_1, \phi^2_1)$

PRIOR SPECIFICATIONS

$p(\beta_0)\ \alpha\ 1$

$p(\beta_1)\ \alpha\ 1$

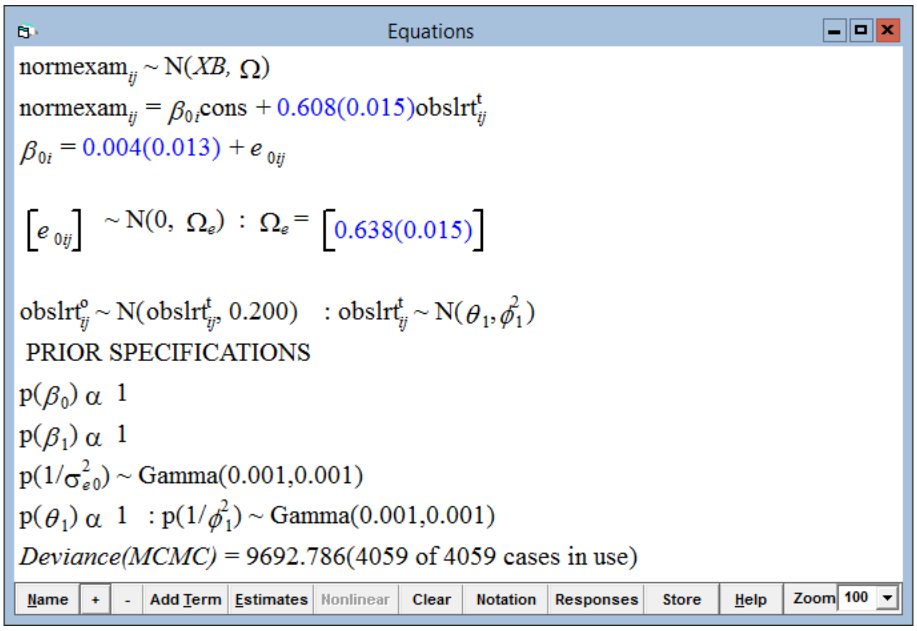
$p(1/\sigma^2_{e0}) \sim \text{Gamma}(0.001,0.001)$

$p(\theta_1)\ \alpha\ 1\quad : p(1/\phi^2_1) \sim \text{Gamma}(0.001,0.001)$

(4059 of 4059 cases in use)

Here we are fitting what is known as a 'classical' Bayesian measurement error model. Basically this includes two additional model statements: firstly that the observed values are Normally distributed around the true values, and secondly that the true values have a Normal distribution with an unknown mean and variance. The first statement by itself would simply have the effect of introducing more random measurement errors to the model, and therefore it is the second statement of a distribution for the true values that fully specifies the measurement error model.

If we now click on the **Start** button to run the model we will see after 5,000 iterations the following results:



$\text{normexam}_{ij} \sim \text{N}(XB, \Omega)$

$\text{normexam}_{ij} = \beta_{0i}\text{cons} + 0.608(0.015)\text{obslrt}^{t}_{ij}$

$\beta_{0i} = 0.004(0.013) + e_{0ij}$

$\begin{bmatrix} e_{0ij} \end{bmatrix} \sim \text{N}(0,\ \Omega_e)\ :\ \Omega_e = \begin{bmatrix} 0.638(0.015) \end{bmatrix}$

$\text{obslrt}^{o}_{ij} \sim \text{N}(\text{obslrt}^{t}_{ij}, 0.200)\quad : \text{obslrt}^{t}_{ij} \sim \text{N}(\theta_1, \phi^2_1)$

PRIOR SPECIFICATIONS

$\text{p}(\beta_0)\ \alpha\ 1$

$\text{p}(\beta_1)\ \alpha\ 1$

$\text{p}(1/\sigma^2_{e0}) \sim \text{Gamma}(0.001,0.001)$

$\text{p}(\theta_1)\ \alpha\ 1\ \ : \text{p}(1/\phi^2_1) \sim \text{Gamma}(0.001,0.001)$

Deviance(MCMC) = 9692.786(4059 of 4059 cases in use)

We can see that the measurement error model has had the desired effects in that the coefficient for the LRT variable has increased and the residual variance has reduced. As this is a simulated dataset (and the errors had a positive relationship with the response) it should be noted that we will not obtain exactly the estimates from fitting the true predictor. However if we run many simulations then, on average, we will get the same predictor values (see Browne et al., 2001*b*, for a multilevel example of this). It should also be noted that the standard error of our coefficient is bigger than when we fit the true predictor and this is due to the measurement error variance. This means that measurement error models will remove the bias in the parameters and allow valid inferences. Measurement error modelling is therefore useful in the main as a tool for sensitivity analysis.

To consider the effect of various measurement errors and hence reliabilities we generated several more sets of errors (all from seed 1). The table below gives the estimates before and after adjusting for measurement error.

M.Error variance	Reliability	β_1 with error	σ_e^2 with error	β_1 after adjustment	σ_e^2 after adjustment
0.1	90.9%	0.549(0.012)	0.673(0.015)	0.605(0.014)	0.641(0.015)
0.2	83.3%	0.505(0.013)	0.699(0.016)	0.608(0.015)	0.638(0.015)
0.5	66.7%	0.404(0.011)	0.757(0.017)	0.612(0.018)	0.633(0.018)
0.7	58.8%	0.357(0.011)	0.784(0.017)	0.614(0.021)	0.632(0.019)
1.0	50.0%	0.303(0.010)	0.816(0.018)	0.614(0.025)	0.630(0.022)

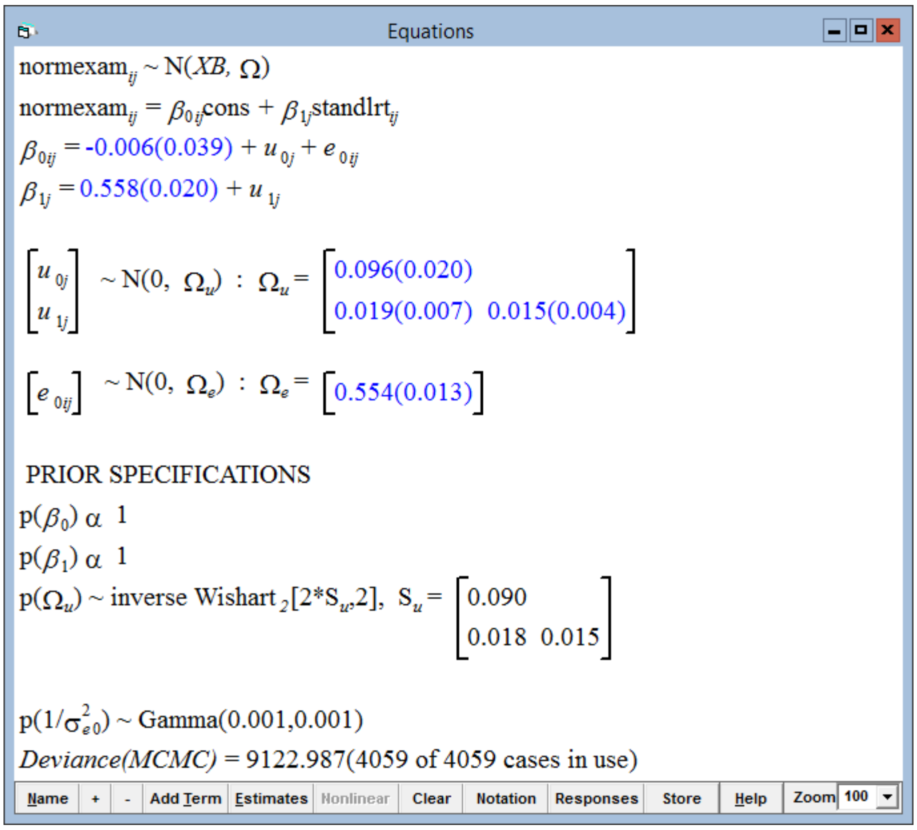
Interestingly the bias induced (for this dataset) in the point estimates increases as the errors increase, and this is due to the slight positive correlation between the errors and the response. The standard errors for the parameters also increase as we are less sure of our estimates the more measurement error we have.

14.2 Measurement error modelling in multilevel models

Browne et al. (2001*b*) consider the effects of measurement errors in a Normal response random slopes regression model. We will now consider this model and firstly set up the model with the true predictor **standlrt**.

- Change **Estimation method** to **IGLS**.
- In the **Equations** window click on β_1 (**obslrt**).
- Change **obslrt** to **standlrt** from the pull down list.
- Click on the **j(school)** tickbox and click on the **Done** button.
- Click on the β_0 (**cons**) predictor in the **Equations** window.
- Click on the **j(school)** tickbox and click on the **Done** button.
- Click on the **Start** button.
- Change **Estimation method** to **MCMC**.
- At this point you may have to remove the measurement errors and this is done by removing the tick in the **Measurement Error** window available from the **Model** menu.
- Click on the **Start** button.

You should now have fitted (using MCMC) the random slopes regression model that we fitted in Chapter 6 and the estimates should be as shown below:



$$\text{normexam}_{ij} \sim \text{N}(XB, \Omega)$$
$$\text{normexam}_{ij} = \beta_{0ij}\text{cons} + \beta_{1j}\text{standlrt}_{ij}$$
$$\beta_{0ij} = -0.006(0.039) + u_{0j} + e_{0ij}$$
$$\beta_{1j} = 0.558(0.020) + u_{1j}$$
$$\begin{bmatrix} u_{0j} \\ u_{1j} \end{bmatrix} \sim \text{N}(0, \Omega_u) : \Omega_u = \begin{bmatrix} 0.096(0.020) & \\ 0.019(0.007) & 0.015(0.004) \end{bmatrix}$$
$$\begin{bmatrix} e_{0ij} \end{bmatrix} \sim \text{N}(0, \Omega_e) : \Omega_e = \begin{bmatrix} 0.554(0.013) \end{bmatrix}$$

PRIOR SPECIFICATIONS

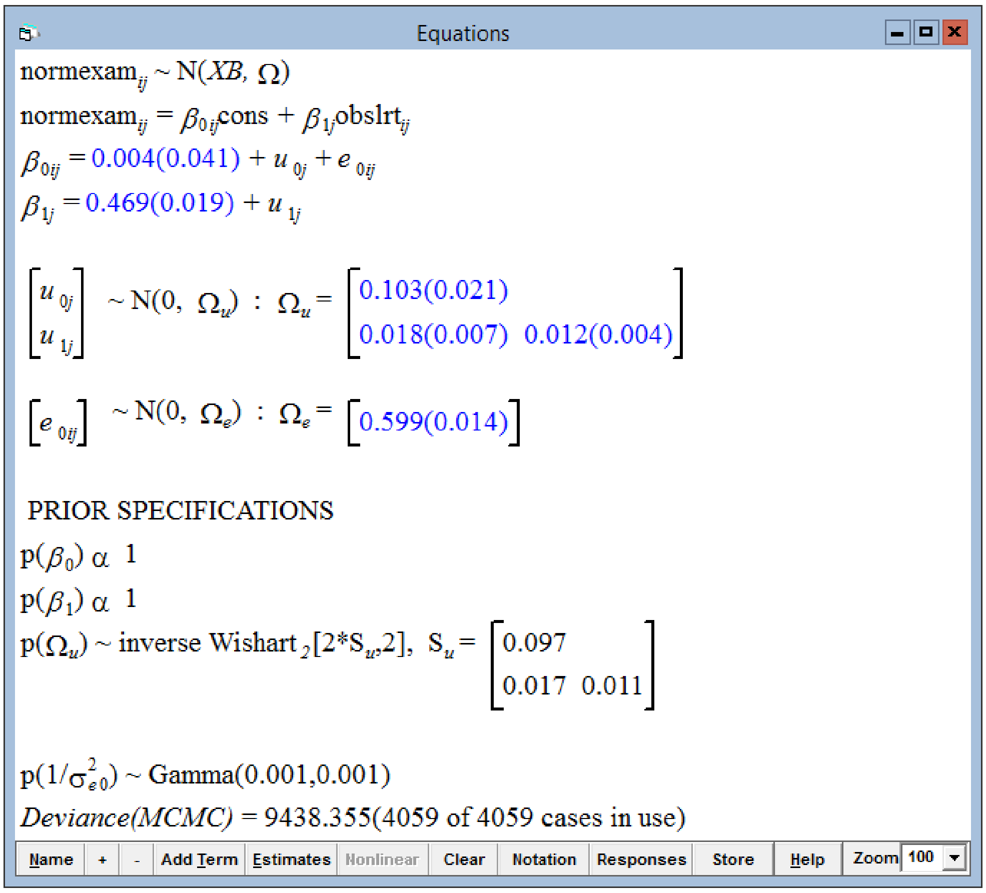
$$p(\beta_0) \propto 1$$
$$p(\beta_1) \propto 1$$
$$p(\Omega_u) \sim \text{inverse Wishart}_2[2*S_u,2], \; S_u = \begin{bmatrix} 0.090 & \\ 0.018 & 0.015 \end{bmatrix}$$
$$p(1/\sigma^2_{e0}) \sim \text{Gamma}(0.001,0.001)$$

Deviance(MCMC) = 9122.987(4059 of 4059 cases in use)

Now if we want to instead use the predictor with measurement error variance = 0.2 (note if you tried out the other measurement error specifications you will need to recreate the **errors** column and the **obslrt** columns) we need to do the following:

- Change **Estimation** method to **IGLS**.
- Click on β_1 (**standlrt**) and change **standlrt** to **obslrt** in the pull down list.
- Untick and retick the **j(school)** box to make sure the random slope is kept in the model.
- Click on the **Done** button.
- Click on the **Start** button.
- Change **Estimation method** to **MCMC**.
- Click on the **Start** button.

Upon running for 5,000 iterations the estimates (ignoring measurement errors) are as follows:


Equations

$\text{normexam}_{ij} \sim \text{N}(XB, \Omega)$

$\text{normexam}_{ij} = \beta_{0ij}\text{cons} + \beta_{1j}\text{obslrt}_{ij}$

$\beta_{0ij} = 0.004(0.041) + u_{0j} + e_{0ij}$

$\beta_{1j} = 0.469(0.019) + u_{1j}$

$$\begin{bmatrix} u_{0j} \\ u_{1j} \end{bmatrix} \sim \text{N}(0, \Omega_u) : \Omega_u = \begin{bmatrix} 0.103(0.021) & \\ 0.018(0.007) & 0.012(0.004) \end{bmatrix}$$

$$\begin{bmatrix} e_{0ij} \end{bmatrix} \sim \text{N}(0, \Omega_e) : \Omega_e = \begin{bmatrix} 0.599(0.014) \end{bmatrix}$$

PRIOR SPECIFICATIONS

$p(\beta_0) \propto 1$

$p(\beta_1) \propto 1$

$$p(\Omega_u) \sim \text{inverse Wishart}_2[2^*S_u, 2], \; S_u = \begin{bmatrix} 0.097 & \\ 0.017 & 0.011 \end{bmatrix}$$

$p(1/\sigma^2_{e0}) \sim \text{Gamma}(0.001, 0.001)$

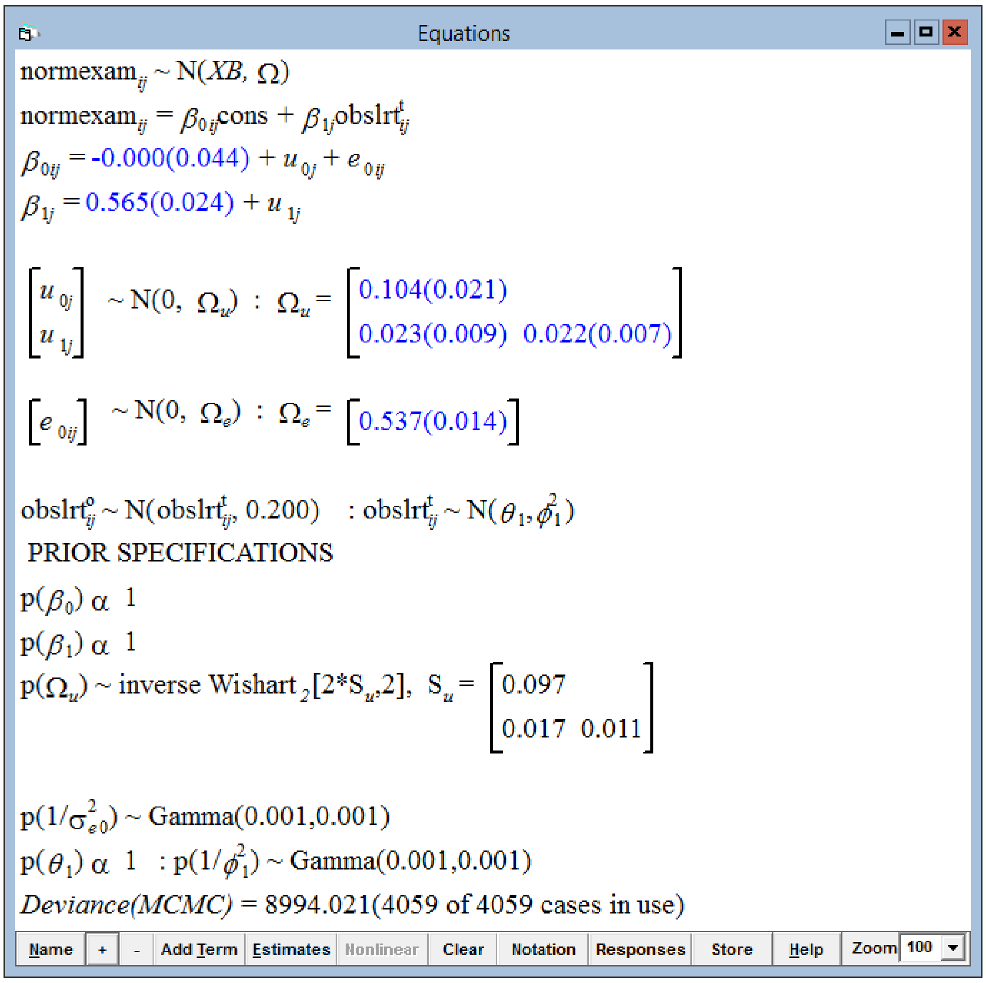
Deviance(MCMC) = 9438.355(4059 of 4059 cases in use)

Here we see that adding measurement error increases the level one variance and reduces the fixed effect associated with intake score, as in the linear regression example. At level two the slopes (intake score) variance is reduced. This makes sense as the regression effects for each school will be reduced in line with the fixed effect and hence the variance will decrease. The intercept variance at level two has increased and this is also to be expected as this will maintain the ratio of the intercept variances at levels one and two.

If we now wish to adjust for measurement error we need simply to set up the errors as before and use MCMC. The measurement error settings will need to be set up as for the linear regression example earlier. To run the method

- Change **Estimation Method** to **IGLS**.
- Click on the **Start** button.
- Change **Estimation Method** to **MCMC**.
- On the **Measurement Errors** window click in the **obslrt** tickbox.
- Input the value **0.2** for the variance.
- Click on the **Done** button
- Click on the **Start** button

After the method runs for 5,000 iterations we will get the following results:



$\text{normexam}_{ij} \sim \text{N}(XB, \Omega)$

$\text{normexam}_{ij} = \beta_{0ij}\text{cons} + \beta_{1j}\text{obslrt}^{t}_{ij}$

$\beta_{0ij} = -0.000(0.044) + u_{0j} + e_{0ij}$

$\beta_{1j} = 0.565(0.024) + u_{1j}$

$$\begin{bmatrix} u_{0j} \\ u_{1j} \end{bmatrix} \sim \text{N}(0,\ \Omega_u)\ :\ \Omega_u = \begin{bmatrix} 0.104(0.021) & \\ 0.023(0.009) & 0.022(0.007) \end{bmatrix}$$

$$\begin{bmatrix} e_{0ij} \end{bmatrix} \sim \text{N}(0,\ \Omega_e)\ :\ \Omega_e = \begin{bmatrix} 0.537(0.014) \end{bmatrix}$$

$\text{obslrt}^{o}_{ij} \sim \text{N}(\text{obslrt}^{t}_{ij}, 0.200)\quad : \text{obslrt}^{t}_{ij} \sim \text{N}(\theta_1, \phi^2_1)$

PRIOR SPECIFICATIONS

$\text{p}(\beta_0) \propto 1$

$\text{p}(\beta_1) \propto 1$

$$\text{p}(\Omega_u) \sim \text{inverse Wishart}_2[2 * \text{S}_u, 2],\ \text{S}_u = \begin{bmatrix} 0.097 & \\ 0.017 & 0.011 \end{bmatrix}$$

$\text{p}(1/\sigma^2_{e0}) \sim \text{Gamma}(0.001, 0.001)$

$\text{p}(\theta_1) \propto 1\ \ : \text{p}(1/\phi^2_1) \sim \text{Gamma}(0.001, 0.001)$

Deviance(MCMC) = 8994.021(4059 of 4059 cases in use)

Here we have used a slightly informative prior at level two based on the estimates of the variance matrix at level 2 from the model that doesn't account for measurement error. This is a slightly different prior than that used in the simulations in Browne et al. (2001*a*) but the same conclusions can be drawn. Here we see that the measurement error model moves all parameters in the right direction apart from the level two intercept variance. Again, due to the positive correlation between response and errors we see that most parameters are over-corrected. In fact Browne et al. (2001*a*) found that the variance parameters at level two in their thousand simulated datasets exhibited positive bias but had near perfect coverage properties. This bias however may be due to the use of the posterior means, rather than modes as point estimates, which tend to give positive biases (see Browne & Draper, 2000).

14.3 Measurement errors in binomial models

As the restriction to continuous measurement error affects only the predictors and not the response it is possible to allow for measurement errors in binary response and Poisson models as easily as in Normal models. That said the additional steps in the MCMC algorithm to update the true values of the predictor will now need to be done using Metropolis rather than Gibbs sampling although this happens automatically and so the user does not need to worry about this.

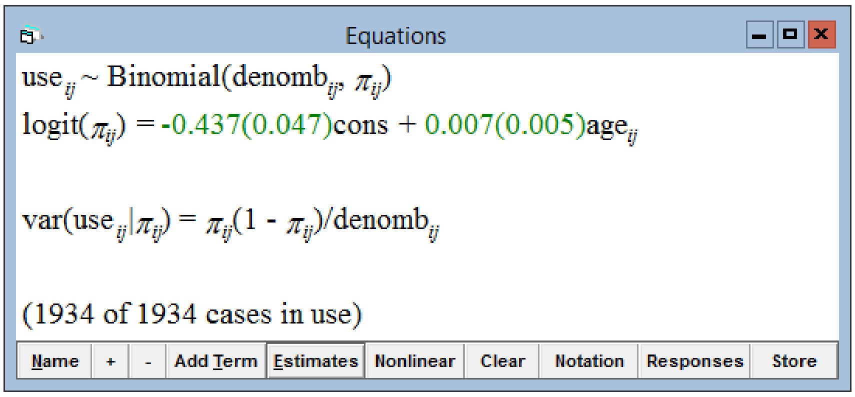
We will consider the first simple example from Chapter 10:

- Select **Open Sample Worksheet** from the **File** menu.
- Select **bang1.ws** from the list and click on **Open**.

We will now set up the simple single level logistic model with **age** as a predictor for contraceptive use.

- Select **Equations** from the **Model** menu.
- Click on the red y.
- Select **use** for the **y** variable.
- Select **2-ij** for the number of levels.
- Select **district** as **level 2(j)**.
- Select **woman** as **level 1(i)**.
- Click on the **Done** button.
- Click on the **N** and instead choose **Binomial** from the list.
- Click on the **Done** button.
- Click on the red x_0.
- Select **cons** and click on the **Done** button.
- Click on the **Add Term** button.
- Select **age** from the **variable** list and click on the **Done** button.
- Click on the red n_{ij} and select **denomb** from the list.
- Click on the **Done** button.
- Click on the **Start** button.

The estimates (from IGLS) are as follows:



$\text{use}_{ij} \sim \text{Binomial}(\text{denomb}_{ij}, \pi_{ij})$

$\text{logit}(\pi_{ij}) = -0.437(0.047)\text{cons} + 0.007(0.005)\text{age}_{ij}$

$\text{var}(\text{use}_{ij}|\pi_{ij}) = \pi_{ij}(1 - \pi_{ij})/\text{denomb}_{ij}$

(1934 of 1934 cases in use)

The predictor to which we will add measurement error is **age**. If we use the **Average and Correlations** window we will get the following summary statistics for **age**:

	N	Missing	Mean	s.d.
age	1934	0	0.0020481	9.0134

We will add measurement errors to **age** with a variance of 25 (s.d. of 5) which will correspond to a reliability of 81.25/(81.25+25) = 76.4%.

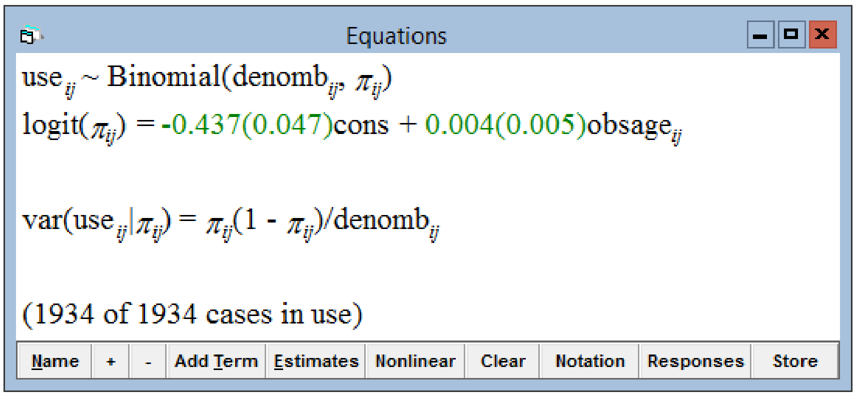
To create the measurement errors we need to type the following commands in the **Command Interface** window:

```
▶ Seed 3
▶ Nran 1934 c16
▶ Calc c16 = c16 * 5
▶ Name c16 'errors'
▶ Calc c17 = 'age' + 'errors'
▶ Name c17 'obsage'
```

Here we have chosen random number seed 3 and this gives simulated errors that have an average of slightly less than zero. Now to see the effect of the errors we need to change the predictor in the regression from **age** to **obsage**.

- Click on the β_1 (**age**) in the **Equations** window.
- Replace **age** with **obsage** from the pull down list.
- Click on the **Done** button.
- Click on the **Start** button.

Upon convergence we get the following results:



$\text{use}_{ij} \sim \text{Binomial}(\text{denomb}_{ij}, \pi_{ij})$

$\text{logit}(\pi_{ij}) = -0.437(0.047)\text{cons} + 0.004(0.005)\text{obsage}_{ij}$

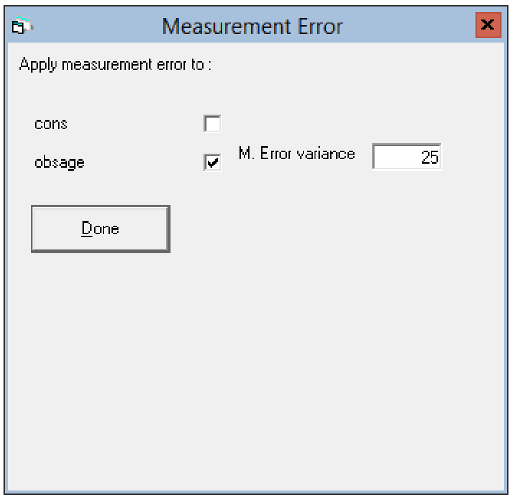
$\text{var}(\text{use}_{ij}|\pi_{ij}) = \pi_{ij}(1 - \pi_{ij})/\text{denomb}_{ij}$

(1934 of 1934 cases in use)

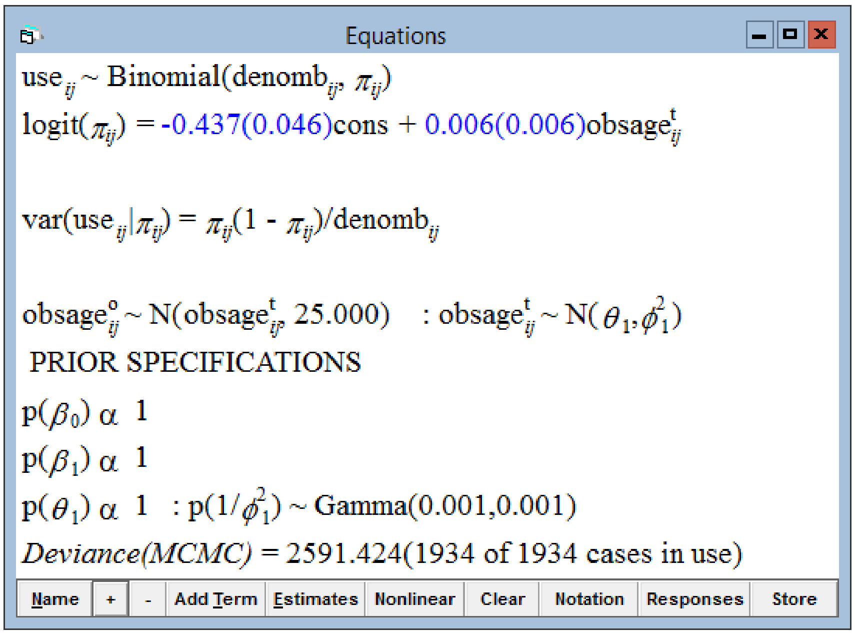
So the errors have reduced the coefficient from 0.007 to 0.004. If we now wish to use MCMC and adjust for the measurement errors we need to do the following:

- Change **Estimation method** to **MCMC**.
- Select **MCMC/Measurement Errors** from the **Model** menu.
- Click on the tick box for **obsage** on the **Measurement Error** window and input **25** as the error variance.

The window should then look as follows:



Now click on the **Done** and **Start** buttons and after a few minutes we will get the following results:


Equations
Name + - Add Term Estimates Nonlinear Clear Notation Responses Store

$\text{use}_{ij} \sim \text{Binomial}(\text{denomb}_{ij}, \pi_{ij})$

$\text{logit}(\pi_{ij}) = -0.437(0.046)\text{cons} + 0.006(0.006)\text{obsage}^t_{ij}$

$\text{var}(\text{use}_{ij}|\pi_{ij}) = \pi_{ij}(1 - \pi_{ij})/\text{denomb}_{ij}$

$\text{obsage}^o_{ij} \sim \text{N}(\text{obsage}^t_{ij}, 25.000) \quad : \text{obsage}^t_{ij} \sim \text{N}(\theta_1, \phi^2_1)$

PRIOR SPECIFICATIONS

$p(\beta_0) \propto 1$

$p(\beta_1) \propto 1$

$p(\theta_1) \propto 1 \quad : p(1/\phi^2_1) \sim \text{Gamma}(0.001,0.001)$

Deviance(MCMC) = 2591.424(1934 of 1934 cases in use)

So we see that the method moves the coefficient for age in the right direction, although the estimate is still slightly lower than the estimate with the true predictor. This is because the errors had a slight negative bias and, given the coefficient for age is small and non-significant to start with, this small bias becomes more important.

14.4 Measurement errors in more than one variable and misclassifications

To finish this chapter we will consider other aspects of measurement error modelling that are available in MLwiN or currently being researched. Firstly if we have many predictors in our model we may have measurement errors on any or all of these predictors. We can account for these errors by adding independent error assumptions for each predictor via the **Measurement Error** window. It is, however, plausible that measurement errors in two predictors could be correlated, although accounting for correlated errors is not currently available in MLwiN. Here the user would have to input both measurement error variances for the predictors and correlations between them and a multivariate Normal distribution would be assumed for the errors.

As we mentioned earlier we are restricted to specifying continuous measurement errors. If however we had a predictor that was a binary outcome, for example a pass/fail intake examination mark, we could theoretically use a continuous error for this predictor. Although this seems rather unusual, a similar argument to that used for the probit modelling in Chapter 10 could be used here.

More generally for binary predictors and categorical predictors we would have to use a misclassification model rather than a measurement error model. For example, consider a predictor variable that has three categories and was fitted into our model as two dummy variable predictors. Then we could consider generating a prior distribution for the probability of observing category A when the true category is category B. We could work out probabilities for all nine combinations of observed and true category where the three probabilities for each true category would sum to 1. We would then have to invert the table of probabilities to produce probabilities of an observation actually being category B when category A is observed. Here the three probabilities associated with category A being observed would need to sum to 1. Using this as a prior distribution we could combine this with the observed data to create a set of posterior probabilities for the true category for each observation, at each iteration. This is harder to program as the dummy variables will have to be modified simultaneously.

Finally we should reiterate that we recommend that measurement error modelling is used primarily as a sensitivity analysis tool. Once measurement errors have been added to a predictor we cannot expect to recover the true predictor but given an estimate of the variance of the errors we can hope to gain a handle on the magnitude of the true predictor coefficients. It should be noted that currently measurement errors cannot be used in MLwiN on parameters that are involved in complex level 1 variance functions (see Chapter 9) or in multivariate models (see Chapters 18–20).

Chapter learning outcomes

- ⋆ What the effects of measurement errors are on predictors.
- ⋆ How to account for measurement errors using MCMC.
- ⋆ How to account for measurement errors in various multilevel models.
- ⋆ How one would in theory account for misclassifications in categorical predictors.

Chapter 15

Cross Classified Models

One of the main uses of multilevel modelling is to account for the underlying structure in a dataset, whether it be pupils nested within schools or women nested within communities as seen in the examples so far. In accounting for the structure we are removing the independence assumption between level one units from the same level two units and instead partitioning the variance into variances between the units at the various levels in the dataset. The examples we have looked at so far have mainly concentrated on two-level structures but we have considered one three level structure (counties within regions within nations) in Chapter 11.

Historically most multilevel modelling has assumed a hierarchical or nested structure for two reasons. Firstly many applications naturally have a nested structure, for example pupils within classes within schools, or patients within wards within hospitals. Secondly the maximum likelihood based methods, for example IGLS, have been designed to work well for nested structures, as fast inversion routines are available for the block diagonal matrices that nested structures produce. However, as we will see in the next three chapters, often the structure of the dataset is not strictly nested. In this chapter we will consider cross-classified models before considering multiple membership models (Chapter 16) and spatial models (Chapter 17).

When cross-classified and multiple membership effects are combined we can produce multiple membership multiple classification (MMMC) models which are described in detail in Browne, Goldstein & Rasbash (2001*a*). Detailed descriptions of likelihood-based methods for both cross-classified models and multiple membership models are given in Rasbash & Goldstein (1994) and Rasbash & Browne (2001), while Rasbash & Browne (2002) compare the likelihood approaches with the MCMC approach that we use here. In this chapter we will describe what we mean by a classification and a cross-classified model before considering an education-based example from Fife, Scotland that is considered in Rasbash et al. (2008, chap. 18).

15.1 Classifications and levels

We have so far concentrated on different ‘levels’ in a dataset where the definition of a level has not been explicitly given, but we have been assuming a nested relationship between levels. For example in education we may have a three ‘level’ dataset with our three levels being pupil, class and school. Here pupils are nested within classes and classes are nested within schools. This implies that all pupils in the same class are also in the same school due to the nesting of the levels. The response variable will be at the lowest level in the dataset although predictors may be at the higher levels, for example the effect of class size on individual pupil scores.

Note that if the response was at a higher level than some of the predictors then these predictors could only be fitted in the model as aggregates. For example we may have several previous tests scores for each pupil, which would imply a lower level of time/test below pupil. If our response was exam score at 16 then we would either fit each previous test as a separate predictor or fit an average previous test mark, and so for the model the lowest level is pupil and not test.

In this chapter we will consider the more general definition of a classification. Having defined our lowest level in the data as the level at which the response variable is collected then we can define a *classification* mathematically as a function, c, that maps from the set θ of N lowest level units to a set Φ of size M where $M \leq N$, and we define the resulting set Φ of M objects as the *classification units*. In this chapter we will only consider single membership classifications, $c(n_i) = m_j, \forall n_i \in \theta$ where $m_j \in \Phi$.

In words, if we consider the educational example earlier then our lowest level was **pupil** and the lowest level units are the individual pupils. Both **school** and **class** will then be classifications (functions) that given an individual pupil will return their respective school and class, and so the sets of all schools and all classes will be the classification units associated with the classifications **school** and **class** respectively. Note that as these classifications map directly from the lowest level there is no guarantee that the classifications will be nested, and in fact nested classifications are a special case of the general ‘cross-classified’ classifications that we consider in this chapter.

MCMC methods treat each set of classification units (residuals in the model) as an additive term in the model and hence it is no more complicated (once the classifications have been calculated) to fit a cross-classified model than a nested model using MCMC. However there is one restriction and that is that we need unique classification identifiers. For example if we truly have a three-level nested model with class 1 in school 1 and class 1 in school 2, then these two classes will need unique identifiers if this model is fitted as a cross-classified model to differentiate between the two class 1s.

15.2 Notation

Browne, Goldstein & Rasbash (2001*a*) introduce notation for fitting cross-classified and more complex models based upon the definition of a classification. Rather than trying to introduce more complex indices that take account of the crossings and nestings (as in Rasbash & Browne, 2001) they instead simply give the response variable subscript i to index lowest level units, and then use the classification names for the subscripts of random effects. For example consider the variance components model described first in Chapter 3. This was written there as:

$$\begin{aligned}\text{normexam}_{ij} &\sim \text{N}(XB, \Omega)\\ \text{normexam}_{ij} &= \beta_{0ij}\text{cons} + \beta_1\text{standlrt}_{ij}\\ \beta_{0ij} &= \beta_0 + u_{0j} + e_{0ij}\end{aligned}$$

In the classification notation we would rewrite this as:

$$\begin{aligned}\text{normexam}_{i} &\sim \text{N}(XB, \Omega)\\ \text{normexam}_{i} &= \beta_{0i}\text{cons}_i + \beta_1\text{standlrt}_{i}\\ \beta_{0i} &= \beta_0 + u^{(2)}_{0,\text{school}(i)} + e_{0i}\end{aligned}$$

As there may be many classifications, rather than using different letters for each, we give a superscript to represent the classification number (note this starts at 2 as we consider the lowest level as classification 1). To change between notations we can use the **Notation** button on the **Equations** window that we earlier used for the alternative complex level 1 notation. We will now consider a cross-classified example from the educational literature.

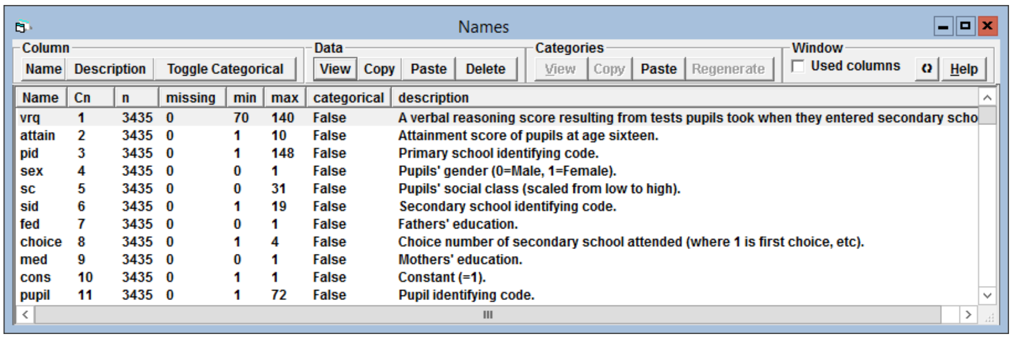
15.3 The Fife educational dataset

We will consider here an educational example from Fife in Scotland that is also considered in the User's Guide to MLwiN (Chapter 18). The data consist of pupils' overall exam attainment at age 16 (as with the **tutorial.ws** dataset studied earlier) and several predictor variables, including a verbal reasoning test taken at age 11 and information on social class and parent's occupation. The added complexity in the dataset is that we have information on both the secondary school (ages 12 through to 16) in which the children studied and the primary school (ages 5 through to 12) they attended prior to secondary school. Not all the children from a particular primary school will attend the same secondary school so we have two classifications that are crossed rather than nested. The data consists of records for 3,435 children from 148 primary schools and 19 secondary schools.

First we will load the dataset and look at the variable names:

- Select **Open Sample Worksheet** from the **File** menu.
- Select **xc1.ws** from the list and click on the **Open** button.

The **Names** window will appear as follows:

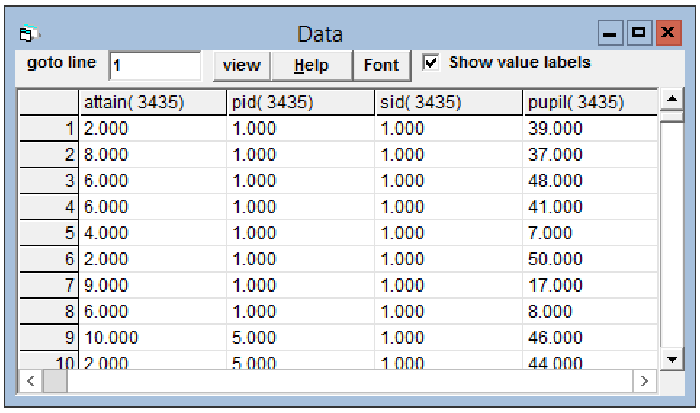


Name	Cn	n	missing	min	max	categorical	description
vrq	1	3435	0	70	140	False	A verbal reasoning score resulting from tests pupils took when they entered secondary scho
attain	2	3435	0	1	10	False	Attainment score of pupils at age sixteen.
pid	3	3435	0	1	148	False	Primary school identifying code.
sex	4	3435	0	0	1	False	Pupils' gender (0=Male, 1=Female).
sc	5	3435	0	0	31	False	Pupils' social class (scaled from low to high).
sid	6	3435	0	1	19	False	Secondary school identifying code.
fed	7	3435	0	0	1	False	Fathers' education.
choice	8	3435	0	1	4	False	Choice number of secondary school attended (where 1 is first choice, etc).
med	9	3435	0	0	1	False	Mothers' education.
cons	10	3435	0	1	1	False	Constant (=1).
pupil	11	3435	0	1	72	False	Pupil identifying code.

We here see that our response variable (**attain**) is a score from 1 to 10 that represents the pupils score on a school leaving exam. The intake ability is measured by a score in a verbal reasoning test, (**vrq**) and we also have predictors that represent gender (**sex**), social class (**sc**), father's education (**fed**), mother's education (**med**) and the choice of secondary school that they attend (**choice** where 1 is first choice and so on).

We can look at the dataset more closely by:

- Select **View or Edit Data** from the **Data Manipulation** menu.
- Select to view columns **attain**, **pid**, **sid** and **pupil**.



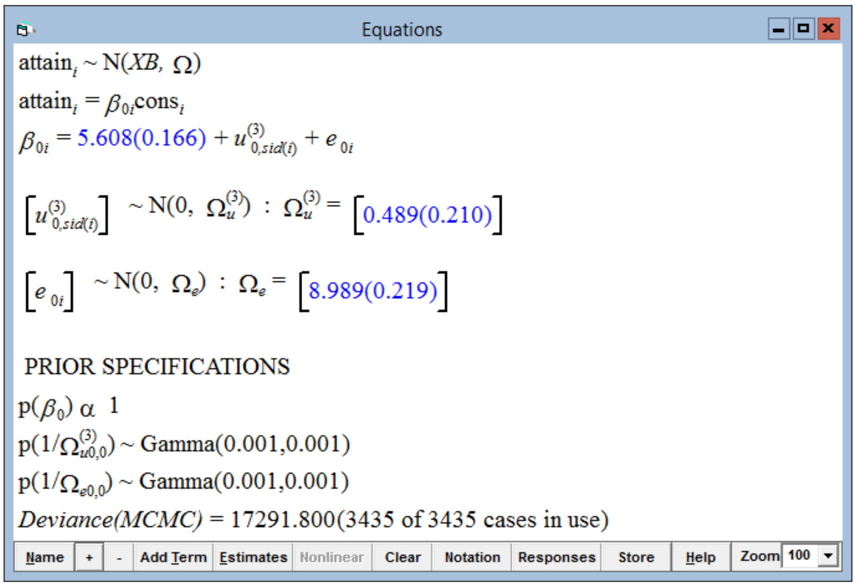
	attain(3435)	pid(3435)	sid(3435)	pupil(3435)
1	2.000	1.000	1.000	39.000
2	8.000	1.000	1.000	37.000
3	6.000	1.000	1.000	48.000
4	6.000	1.000	1.000	41.000
5	4.000	1.000	1.000	7.000
6	2.000	1.000	1.000	50.000
7	9.000	1.000	1.000	17.000
8	6.000	1.000	1.000	8.000
9	10.000	5.000	1.000	46.000
10	2.000	5.000	1.000	44.000

The data have been sorted on primary school within secondary school. We can see here that 8 of the pupils who attended primary school 1 then attended secondary school 1. If we were to scan down the columns we would find that the rest of primary school 1 went to two other secondary schools, 45 to secondary school 9 and 1 to secondary school 18 (to see this quickly type 1355 or 3068 into the **goto line** box and this will take you to these groups of pupils). So we can see that school 9 is the 'main' secondary school for primary school 1 with 83% of pupils attending it. In the entire dataset 59 of the 148 primary schools had all their pupils attend the same secondary school after leaving primary school and only 288 pupils did not attend their 'main' secondary school. So although the dataset structure is not nested it is close to nested and this helps the likelihood-based methods in the User's Guide to MLwiN (see Rasbash & Goldstein, 1994, for details). The degree of 'nestedness' does not matter so much to the MCMC methods and in fact it is probably easier to distinguish between two classifications if they are less nested!

As the data are sorted on secondary schools and their effects will have happened closer (in time) to the exam response of interest we will first consider fitting a two-level model of children within secondary school. We will however use the classification notation from the start and define the three-classification structure of the data.

- Select **Equations** from the **Model** menu.
- Click on the **Notation** button and remove the tick for multiple subscripts and an i subscript will appear on the red y.
- Click on the **Done** button.
- Click on the red y and select **ATTAIN** from the y pull down list.
- Select 3 from the **N classifications** box.
- Select **sid** as classification 3, **PID** as classification 2 and **PUPIL** as classification 1.
- Click on the **Done** button.
- Click on the red x_0 and select **CONS** from the pull down list.
- Select **cons** as a fixed effect and random at classifications **pupil(1)** and **sid(3)**.
- Click on the **Done** button.
- Click on the **Start** button.
- Change **Estimation method** to **MCMC**.
- Click on the **Start** button.

This will have set up the 2 level variance components model and run it using MCMC. The estimates in the **Equations** window will be as follows:



$\text{attain}_i \sim \text{N}(XB, \Omega)$

$\text{attain}_i = \beta_{0i}\text{cons}_i$

$\beta_{0i} = 5.608(0.166) + u^{(3)}_{0,sid(i)} + e_{0i}$

$\left[u^{(3)}_{0,sid(i)}\right] \sim \text{N}(0,\ \Omega^{(3)}_u)\ :\ \Omega^{(3)}_u = \left[0.489(0.210)\right]$

$\left[e_{0i}\right] \sim \text{N}(0,\ \Omega_e)\ :\ \Omega_e = \left[8.989(0.219)\right]$

PRIOR SPECIFICATIONS

$p(\beta_0)\ \alpha\ 1$

$p(1/\Omega^{(3)}_{u0,0}) \sim \text{Gamma}(0.001,0.001)$

$p(1/\Omega_{e0,0}) \sim \text{Gamma}(0.001,0.001)$

Deviance(MCMC) = 17291.800(3435 of 3435 cases in use)

Name + - Add Term Estimates Nonlinear Clear Notation Responses Store Help Zoom 100

Here we see that there is significant variation between the secondary schools and this accounts for $0.489/(0.489+8.989)\times100\% = 5.1\%$ of the total variation in exam marks.

We can compare the DIC for this model with a simpler model with no school effects, and we see a reduction in DIC of 120 showing this is a much better model. Also the 19 secondary school effects account for $18.2 - 2 = 16.2$ effective parameters so there are distinct differences between secondary schools.

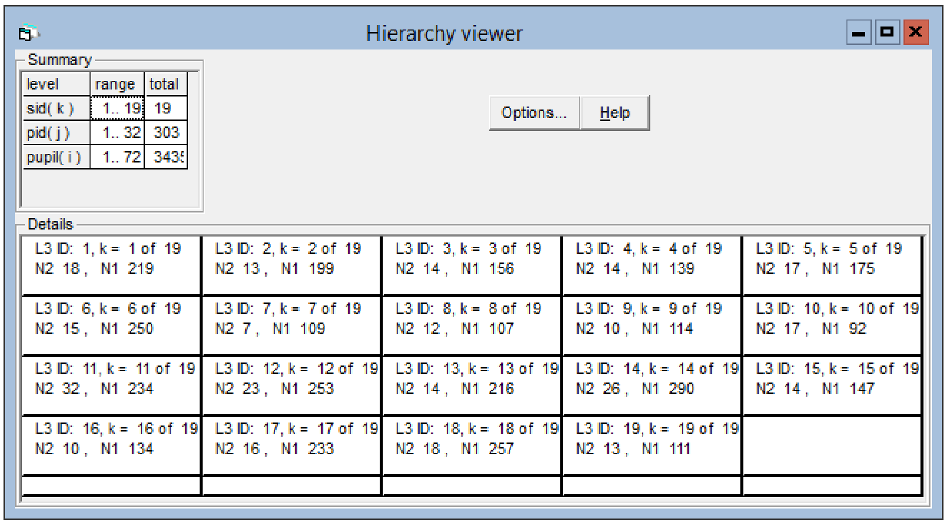
Dbar	D(thetabar)	pD	DIC	
17291.80	17273.61	18.19	17309.99	
17429.27	17427.26	2.01	17431.28	(no school effect)

15.4 A Cross-classified model

If we now consider adding in the effects of primary schools this can be done simply via the **Equations** window.

- Change **Estimation method** to **IGLS**.
- Click on x_0 (**cons**) and tick the **PID(2)** box.
- Click on the **Start** button.

What you have actually just done is fitted a 'nested' model of primary school nested within secondary school using IGLS. This can be confirmed by looking at the **Hierarchy viewer** available via the **Model** menu.



Summary

level	range	total
sid(k)	1.. 19	19
pid(j)	1.. 32	303
pupil(i)	1.. 72	343

Details

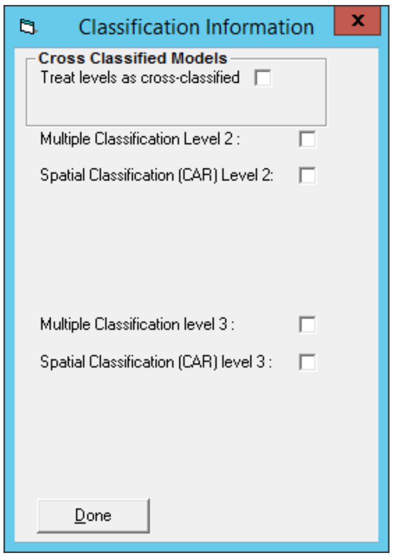
L3 ID: 1, k = 1 of 19 N2 18 , N1 219	L3 ID: 2, k = 2 of 19 N2 13 , N1 199	L3 ID: 3, k = 3 of 19 N2 14 , N1 156	L3 ID: 4, k = 4 of 19 N2 14 , N1 139	L3 ID: 5, k = 5 of 19 N2 17 , N1 175
L3 ID: 6, k = 6 of 19 N2 15 , N1 250	L3 ID: 7, k = 7 of 19 N2 7 , N1 109	L3 ID: 8, k = 8 of 19 N2 12 , N1 107	L3 ID: 9, k = 9 of 19 N2 10 , N1 114	L3 ID: 10, k = 10 of 19 N2 17 , N1 92
L3 ID: 11, k = 11 of 19 N2 32 , N1 234	L3 ID: 12, k = 12 of 19 N2 23 , N1 253	L3 ID: 13, k = 13 of 19 N2 14 , N1 216	L3 ID: 14, k = 14 of 19 N2 26 , N1 290	L3 ID: 15, k = 15 of 19 N2 14 , N1 147
L3 ID: 16, k = 16 of 19 N2 10 , N1 134	L3 ID: 17, k = 17 of 19 N2 16 , N1 233	L3 ID: 18, k = 18 of 19 N2 18 , N1 257	L3 ID: 19, k = 19 of 19 N2 13 , N1 111	

Here you can see that MLwiN has treated the individual groups of pupils that are from the same primary school and secondary school as separate primary schools, for example the pupils in primary school 1 are treated as three separate primary schools nested within secondary schools 1, 9 and 18 respectively. This results in 303 rather than 148 primary schools. To fit a cross-classified model in IGLS instead involves following the procedures given in Chapter 18 of the User's Guide to MLwiN.

To fit the model (as cross-classified) using MCMC is however fairly simple.

- Change **Estimation method** to **MCMC**.
- Select **MCMC/Classifications** from the **Model** menu.

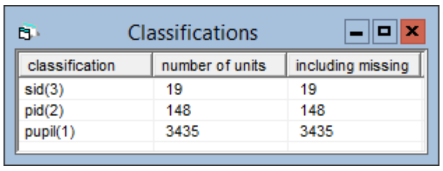
The window will appear as follows:



Cross Classified Models
Treat levels as cross-classified ☐

Multiple Classification Level 2 : ☐
Spatial Classification (CAR) Level 2: ☐
Multiple Classification level 3 : ☐
Spatial Classification (CAR) level 3 : ☐

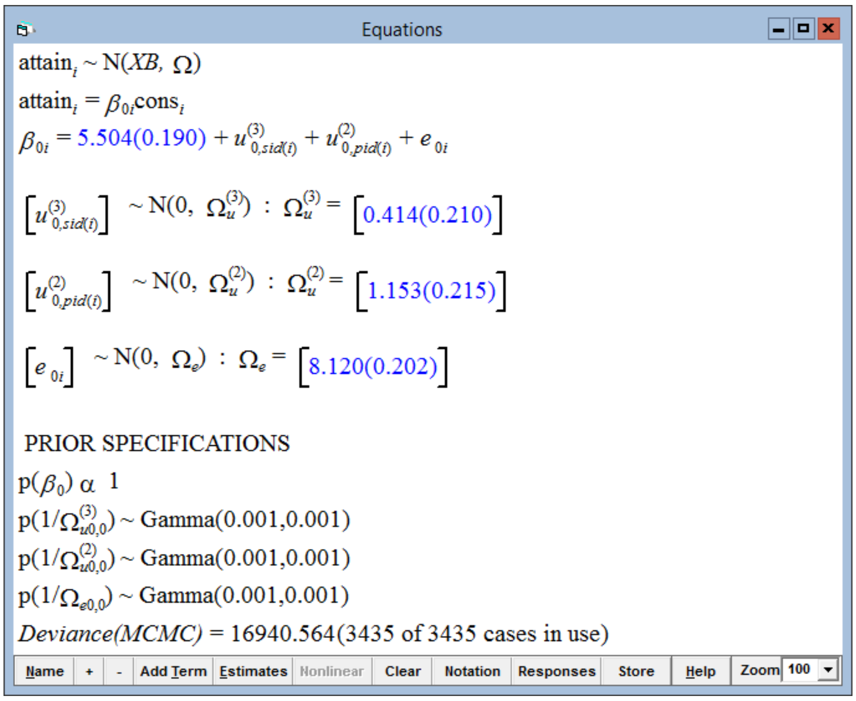
Here we now simply have to click in the **Treat levels as cross-classified** box and click on the **Done** button. If we now select the **Hierarchy Viewer** from the **Model** menu we get the alternative classifications viewer as shown below.



Classifications

classification	number of units	including missing
sid(3)	19	19
pid(2)	148	148
pupil(1)	3435	3435

Here we see that this viewer shows we have only 148 primary schools as we are now taking account of the cross-classifications. After running the model by clicking on the **Start** button we will get the following estimates:



Equations

$$\text{attain}_i \sim \text{N}(XB,\ \Omega)$$

$$\text{attain}_i = \beta_{0i}\text{cons}_i$$

$$\beta_{0i} = 5.504(0.190) + u^{(3)}_{0,sid(i)} + u^{(2)}_{0,pid(i)} + e_{0i}$$

$$\left[u^{(3)}_{0,sid(i)}\right] \sim \text{N}(0,\ \Omega^{(3)}_u)\ :\ \Omega^{(3)}_u = \left[0.414(0.210)\right]$$

$$\left[u^{(2)}_{0,pid(i)}\right] \sim \text{N}(0,\ \Omega^{(2)}_u)\ :\ \Omega^{(2)}_u = \left[1.153(0.215)\right]$$

$$\left[e_{0i}\right] \sim \text{N}(0,\ \Omega_e)\ :\ \Omega_e = \left[8.120(0.202)\right]$$

PRIOR SPECIFICATIONS

$$p(\beta_0)\ \alpha\ 1$$

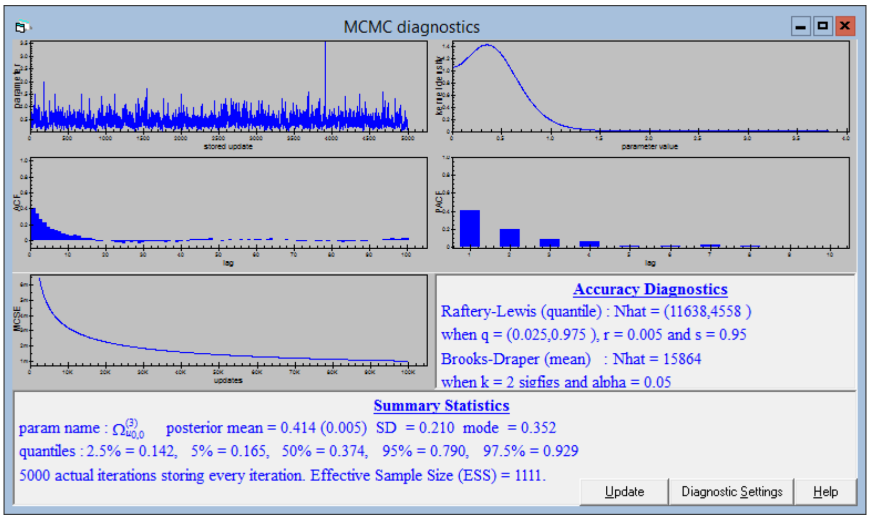
$$p(1/\Omega^{(3)}_{u0,0}) \sim \text{Gamma}(0.001,0.001)$$

$$p(1/\Omega^{(2)}_{u0,0}) \sim \text{Gamma}(0.001,0.001)$$

$$p(1/\Omega_{e0,0}) \sim \text{Gamma}(0.001,0.001)$$

Deviance(MCMC) = 16940.564(3435 of 3435 cases in use)

The estimates are fairly similar to those achieved using IGLS in the User's Guide to MLwiN although the variances for primary school (1.15 versus 1.12) and particularly secondary school (0.41 versus 0.35) are higher. This is due to the difference between mean estimates and mode (ML) estimates for the skewed variance parameter posterior distributions. The trajectory plots confirm this for the secondary school variance:



Accuracy Diagnostics
Raftery-Lewis (quantile) : Nhat = (11638,4558)
when q = (0.025,0.975), r = 0.005 and s = 0.95
Brooks-Draper (mean) : Nhat = 15864
when k = 2 sigfigs and alpha = 0.05

Summary Statistics
param name : $\Omega^{(3)}_{u0,0}$ posterior mean = 0.414 (0.005) SD = 0.210 mode = 0.352
quantiles : 2.5% = 0.142, 5% = 0.165, 50% = 0.374, 95% = 0.790, 97.5% = 0.929
5000 actual iterations storing every iteration. Effective Sample Size (ESS) = 1111.

We can also see that primary school is actually more important in predicting the attainment score than secondary school. One possible reason for this is that secondary schools are generally larger (see Goldstein, 2003). Here primary school explains $1.15/(0.41{+}1.15{+}8.12){\times}100\% = 11.9\%$ of variation while secondary school only explains $0.41/(0.41{+}1.15{+}8.12){\times}100\% = 4.2\%$. The DIC diagnostic again shows that this model is an improvement with a reduction in DIC of over 250.

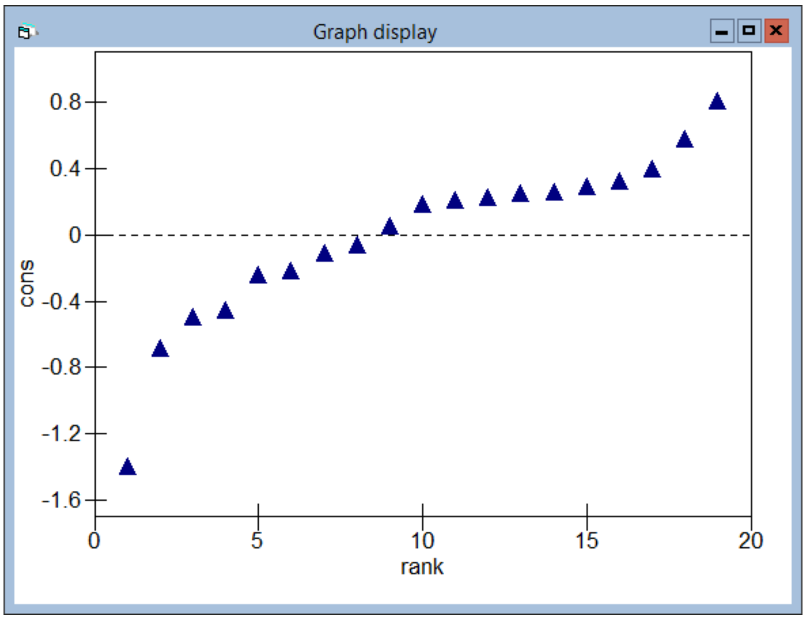
Dbar	D(thetabar)	pD	DIC	
16940.56	16833.40	107.16	17047.73	(with primary school)
17291.80	17273.61	18.19	17309.99	(without primary school)

15.5 Residuals

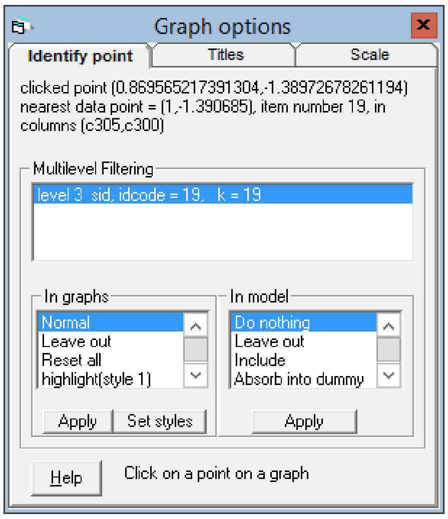
As with nested models we can work out residuals for the various levels of our model. This may be done via the **Residuals** window available from the **Model** menu. We will look firstly at secondary school residuals:

- On the **Residuals** window, change the **level** box to **3:SID**.
- Click on the **Calc** button.
- Click on the **Plots** tab, and if not selected, select **residual x rank**.
- Click on the **Apply** button.

The plot will then appear as follows:



Here we see the lowest ranked secondary school has a very low residual and may be an outlier. Clicking on the graph on this point we get:



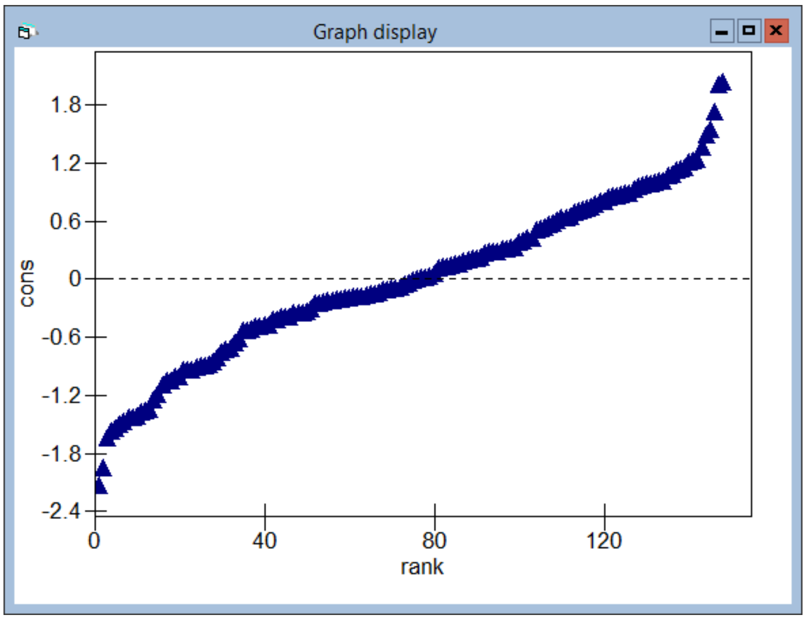
clicked point (0.869565217391304,-1.38972678261194)
nearest data point = (1,-1.390685), item number 19, in columns (c305,c300)

level 3 sid, idcode = 19, k = 19

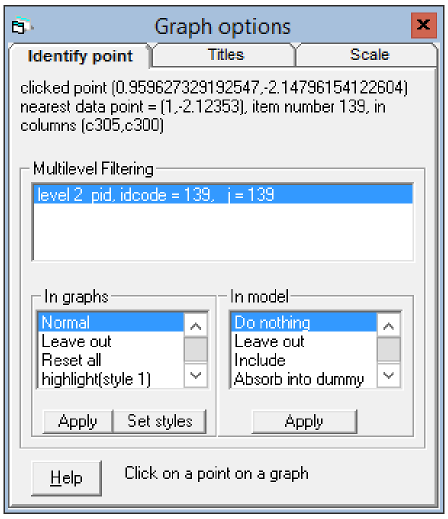
Click on a point on a graph

showing that this is secondary school 19. We will revisit this plot after adding in other variables. If we now look instead at the primary schools:

- On the **Residuals** window, click on the **Settings** tab.
- Change the **level** box to **2:PID**.
- Click on the **Calc** button.
- Click on the **Plots** tab, and if not selected select **residual x rank**.
- Click on the **Apply** button.



Here we see the 148 primary school residuals. Here there is no evidence of outliers. If we click on the lowest residual (rank 1) we get the following:



clicked point (0.959627329192547,-2.14796154122604)
nearest data point = (1,-2.12353), item number 139, in
columns (c305,c300)

level 2 pid, idcode = 139, j = 139

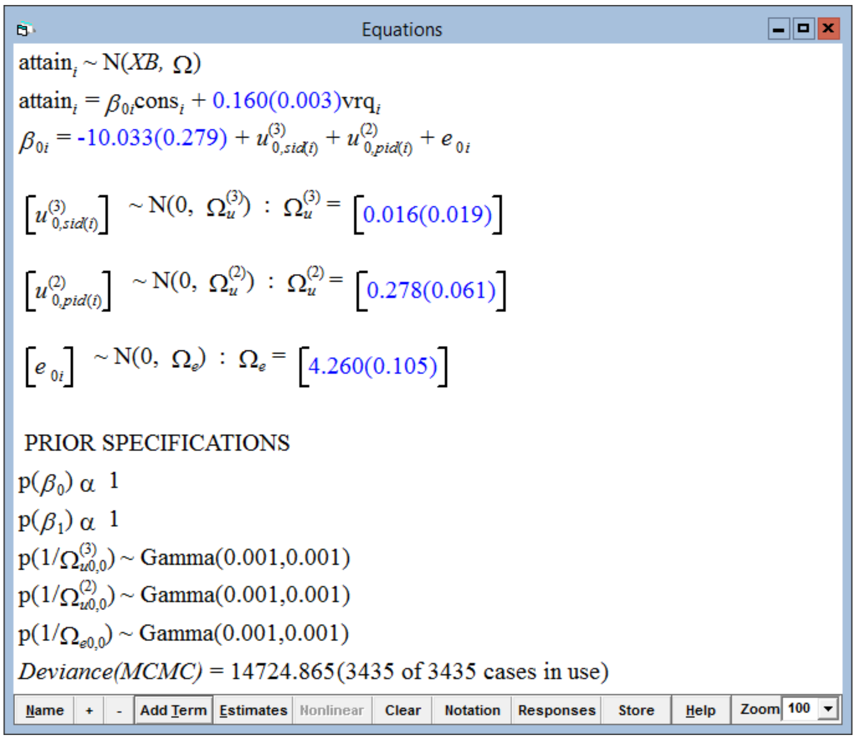
So we see here that the lowest ranked primary school is school number 139 and that even though the data are not nested the residuals screen can identify correctly the primary school. Note however that unlike nested models we do not get a level 3 identifier as primary school is not nested within secondary school.

15.6 Adding predictors to the model

We have so far not considered any of the available predictors in our model. We will firstly consider the effect of intake score (**VRQ**) in our model.

- Change **Estimation method** to **IGLS**.
- Click on the **Add Term** button and select **VRQ** from the **variable** list.
- Click on the **Done** button.
- Click on the **Start** button.
- Change **Estimation method** to **MCMC**.
- Click on the **Start** button.

The estimates produced are as follows:


Equations

$\text{attain}_i \sim \text{N}(XB,\ \Omega)$

$\text{attain}_i = \beta_{0i}\text{cons}_i + 0.160(0.003)\text{vrq}_i$

$\beta_{0i} = -10.033(0.279) + u^{(3)}_{0,sid(i)} + u^{(2)}_{0,pid(i)} + e_{0i}$

$\left[u^{(3)}_{0,sid(i)}\right] \sim \text{N}(0,\ \Omega^{(3)}_u)\ :\ \Omega^{(3)}_u = \left[0.016(0.019)\right]$

$\left[u^{(2)}_{0,pid(i)}\right] \sim \text{N}(0,\ \Omega^{(2)}_u)\ :\ \Omega^{(2)}_u = \left[0.278(0.061)\right]$

$\left[e_{0i}\right] \sim \text{N}(0,\ \Omega_e)\ :\ \Omega_e = \left[4.260(0.105)\right]$

PRIOR SPECIFICATIONS

$p(\beta_0)\ \alpha\ 1$

$p(\beta_1)\ \alpha\ 1$

$p(1/\Omega^{(3)}_{u0,0}) \sim \text{Gamma}(0.001,0.001)$

$p(1/\Omega^{(2)}_{u0,0}) \sim \text{Gamma}(0.001,0.001)$

$p(1/\Omega_{e0,0}) \sim \text{Gamma}(0.001,0.001)$

Deviance(MCMC) = 14724.865(3435 of 3435 cases in use)

The predictor, **vrq**, explains not only a large amount of the residual variation but also a large amount of the differences between secondary schools and between primary schools. Of the remaining variation, 6% is explained by primary schools and less than 0.4% by secondary schools. The DIC diagnostic gives:

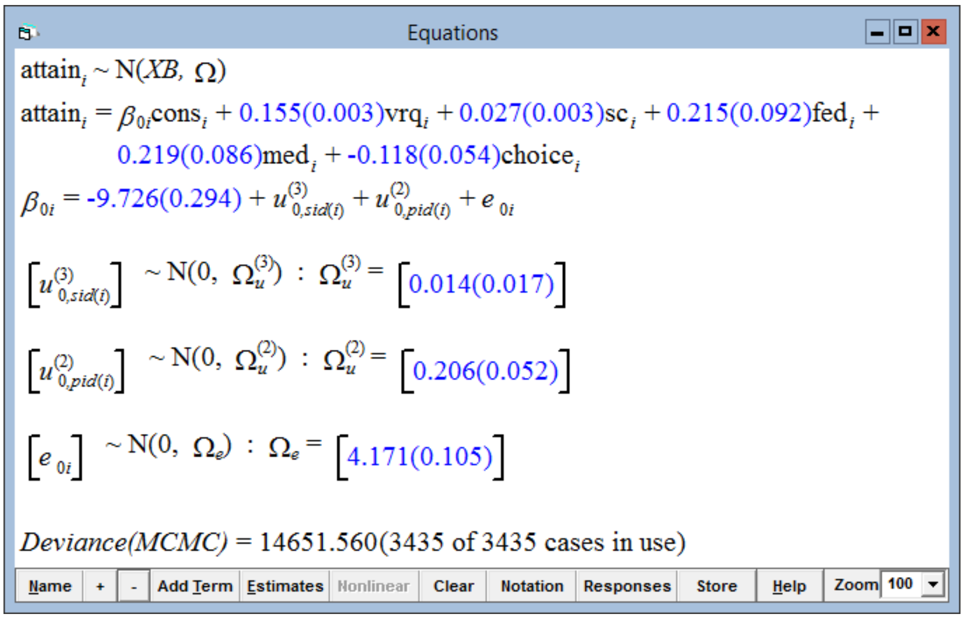
Dbar	D(thetabar)	pD	DIC	
14724.86	14644.21	80.66	14805.52	(with vrq)
16940.56	16833.40	107.16	17047.73	(without vrq)

which shows a reduction in DIC of over 2000! It is also interesting that the effective number of parameters is reduced and this is clearly because VRQ is explaining many of the differences between secondary schools and between primary schools.

We can continue adding in the other predictor variables and retaining significant predictors. In this case all predictors tested apart from gender (**SEX**) are significant. The model with all significant predictors can be obtained by:

- Change **Estimation method** to **IGLS**.
- Add the variables **SC**, **FED**, **MED** and **CHOICE** as fixed effects to the model.
- Click on the **Start** button.
- Change **Estimation method** to **MCMC**.
- Click on the **Start** button.

When the 5,000 iterations have been run we get the following estimates:



$$\text{attain}_i \sim \text{N}(XB, \Omega)$$

$$\text{attain}_i = \beta_{0i}\text{cons}_i + 0.155(0.003)\text{vrq}_i + 0.027(0.003)\text{sc}_i + 0.215(0.092)\text{fed}_i + 0.219(0.086)\text{med}_i + \text{-}0.118(0.054)\text{choice}_i$$

$$\beta_{0i} = \text{-}9.726(0.294) + u^{(3)}_{0,sid(i)} + u^{(2)}_{0,pid(i)} + e_{0i}$$

$$\left[u^{(3)}_{0,sid(i)}\right] \sim \text{N}(0,\ \Omega^{(3)}_u)\ :\ \Omega^{(3)}_u = \left[0.014(0.017)\right]$$

$$\left[u^{(2)}_{0,pid(i)}\right] \sim \text{N}(0,\ \Omega^{(2)}_u)\ :\ \Omega^{(2)}_u = \left[0.206(0.052)\right]$$

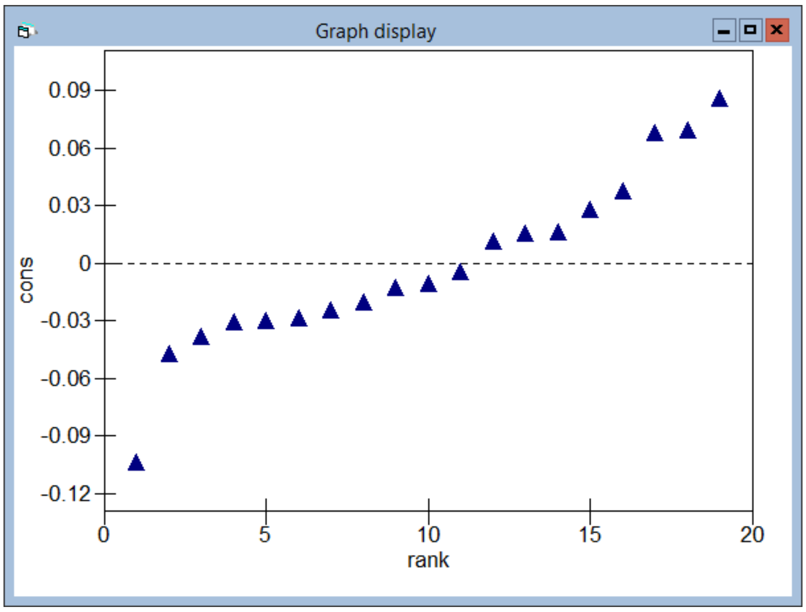
$$\left[e_{0i}\right] \sim \text{N}(0,\ \Omega_e)\ :\ \Omega_e = \left[4.171(0.105)\right]$$

Deviance(MCMC) = 14651.560(3435 of 3435 cases in use)

Here we see that on average a pupil's attainment is higher if they come from a higher social class, if their parents are better educated or if the school they attend is their first choice. Adding the additional predictors has the effect of reducing the DIC diagnostic by 80 and again reducing the effective number of parameters slightly, suggesting more of the differences between schools have been explained by the additional predictors.

Dbar	D(thetabar)	pD	DIC	
14651.56	14575.02	76.54	14728.10	
14724.86	14644.21	80.66	14805.52	(without additional predictors)

The secondary school variance is very small and if we now look at the residuals plot of the school residuals against rank (see instructions earlier on how to produce this) we see that the residual for school 19 is still lowest and looks like an outlier. (Note that a number of error messages may crop up during the estimation here. It is safe to ignore them by clicking the **OK** button.)



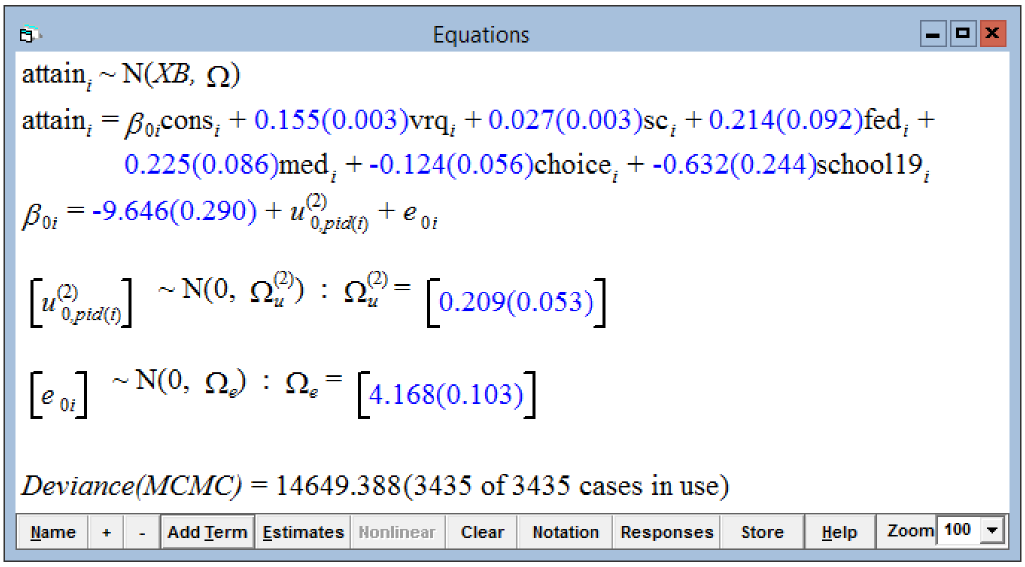
We will therefore consider fitting a dummy variable for school 19 and removing secondary school from the model.

- Select **Command Interface** from the **Data Manipulation** menu and enter the following commands:

```
▶ calc c12 = 'sid' == 19
▶ name c12 'school19'
```

- Change **Estimation method** to **IGLS**.
- Click on the β_0 (**cons**) and remove the tick for **sid(3)**.
- Click on the **Add Term** button and select **school19** from the list.
- Click on the **Done** button.
- Click on the **Start** button.
- Change **Estimation method** to **MCMC**.
- Click on the **Start** button.

After the 5,000 iterations have completed our estimates are as follows:



$$\text{attain}_i \sim \text{N}(XB, \Omega)$$

$$\text{attain}_i = \beta_{0i}\text{cons}_i + 0.155(0.003)\text{vrq}_i + 0.027(0.003)\text{sc}_i + 0.214(0.092)\text{fed}_i + 0.225(0.086)\text{med}_i + \text{-}0.124(0.056)\text{choice}_i + \text{-}0.632(0.244)\text{school19}_i$$

$$\beta_{0i} = \text{-}9.646(0.290) + u^{(2)}_{0,pid(i)} + e_{0i}$$

$$\left[u^{(2)}_{0,pid(i)}\right] \sim \text{N}(0,\ \Omega^{(2)}_u)\ :\ \Omega^{(2)}_u = \left[0.209(0.053)\right]$$

$$\left[e_{0i}\right] \sim \text{N}(0,\ \Omega_e)\ :\ \Omega_e = \left[4.168(0.103)\right]$$

Deviance(MCMC) = 14649.388(3435 of 3435 cases in use)

We can see that school 19 has a significant negative effect on attainment and if we look at the DIC diagnostic we see an improvement in DIC diagnostic of 3.4.

Dbar	D(thetabar)	pD	DIC	
14649.39	14574.03	75.36	14724.74	(with secondary school 19 only)
14651.56	14575.02	76.54	14728.10	(with all secondary school effects)

So in adding the predictors to our model we have explained all the secondary school variation down to a difference between school 19 and the rest of the secondary schools. This of course means that, for the Fife dataset, we now no longer need to fit a cross-classified model. Therefore if we were to re-sort the data on primary school we could have fitted the final model directly using IGLS or MCMC. Some people may think this is disappointing but with only 19 secondary schools to start with it is unlikely that we will find much variation and in fact we now have a more parsimonious model. It may be interesting for the researchers to now go and investigate why school 19 was a potential outlier.

15.7 Current restrictions for cross-classified models

As has been shown in this chapter it is now possible to quite easily fit cross-classified models in MLwiN using MCMC, although not all features have been updated to account for these models. For example currently the **Predictions** window does not account for cross-classified random effects and will therefore give error messages if it is used. It should also be noted that the starting values that MCMC gets for the residuals will be based on the values obtained from the nested model and so will often be meaningless. It is possible by running the MCMC and other commands in the **Command interface** window to fit the separate IGLS two-level models and store these residuals in columns to be used as starting values, but generally the MCMC routines are robust to the nested model starting values. Currently cross-classified models can be fitted using IGLS, but only via additional commands that transform the cross-classified model into a constrained nested model.

Chapter learning outcomes

- ⋆ What is meant by a classification and a cross-classified model.
- ⋆ How to fit cross-classified models in MLwiN using MCMC.
- ⋆ How to look at residuals in a cross-classified model.
- ⋆ Some of the current restrictions in fitting cross-classified models in MLwiN.

Chapter 16

Multiple Membership Models

In the last chapter we considered cross-classified models and introduced the concept of a classification. All the classifications we considered were what we would describe as 'single membership' classifications. This means that every lowest level unit is a member of one and only one classification unit. For example each pupil in the **tutorial.ws** dataset belongs to one and only one school and each woman in the **bang1.ws** dataset belongs to one and only one district.

It is however possible that we cannot (or do not want to) assign each lowest level unit to exactly one classification unit. This may be due to movements between units over the time period for which the data were collected. For example if our response is exam scores at 16 then some pupils will have been educated in more than one school and thus we may want to account for the effects of all schools. Alternatively our response may have been produced by the aggregation of units. For example in veterinary epidemiology the unit of measurement may be flocks of chickens and each flock may be produced from individual parent birds from several parent flocks, each of which has an effect on the child flock (see Browne, Goldstein & Rasbash, 2001*a*, for details). Both of these scenarios are examples of 'multiple membership' classifications.

Formally we can define a multiple membership classification as a map c from the set Θ of N lowest level units to the set Φ of M classification units such that each individual, $n_i \in \Theta$ is mapped to a subset (possibly of size 1) Φ_i of Φ. So the single membership classifications are a special case where every Φ_i is of size 1. We will firstly consider in this chapter a (simulated) example from employment statistics that has a multiple membership classification, and finish the chapter with a look at the combination of multiple membership models and cross-classified models known as multiple membership multiple classification (MMMC) models (Browne et al., 2001*a*).

16.1 Notation and weightings

In the last chapter we introduced a new notation for use with classifications. We can extend this classification to deal with multiple membership classifications. When we have a multiple membership classification, a single lowest level unit will have a random effect for each 'classification unit' they belong to. Generally, to avoid lowest level units that belong to many classification units being given too much influence in the model, we assign weights for each pairing of lowest level unit and classification unit. These weights typically sum to 1 for each lowest level unit and we can write a simple multiple membership model of pupils nested in multiple schools as:

$$\begin{aligned} y_i &\sim \mathrm{N}(XB, \Omega) \\ y_i &= \beta_{0i} x_{0i} \\ \beta_{0i} &= \beta_0 + \sum_{j \in school(i)} w^{(2)}_{i,j} u^{(2)}_{0j} + e_{0i} \end{aligned}$$

Here we have a classification **school** and the weight $w^{(2)}_{i,j}$ is the weight assigned to the random effect for school j in the equation for pupil i. What quantities to use as weights is an interesting question. If we have no information other than that each pupil went to a particular selection of schools over the period of their education then equal weights would be logical. If however we know how long they spent in each school then we could use this information to create weights proportional to the times spent in each school.

Both of these methods are making an assumption that the effect of a school is some fraction of the amount of time spent in all schools by each pupil and hence the weights for each pupil sum to 1. An alternative approach would be that schools have an instantaneous effect that is equal for all pupils no matter how long they spend there. We could of course fit this instantaneous effect by having weights of one for each pupil by school combination but this will make comparison of the between schools variance and residual variance difficult.

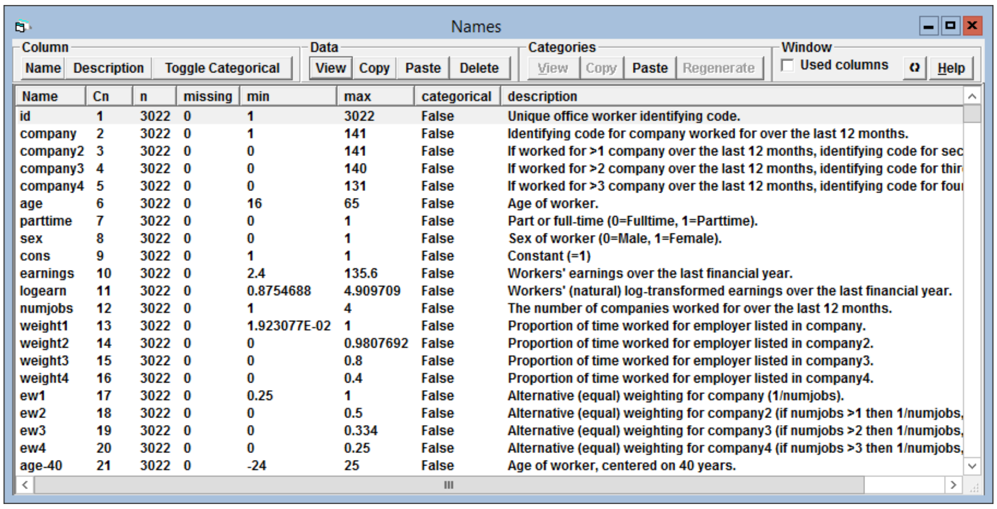
16.2 Office workers salary dataset

We will consider as an example a simulated dataset meant to represent the yearly salaries of randomly sampled office workers.

- Select **Open Sample Worksheet** from the **File** menu.

- Select **wage1.ws** from the list and click on the **Open** button.

The **Names** window will appear as follows:



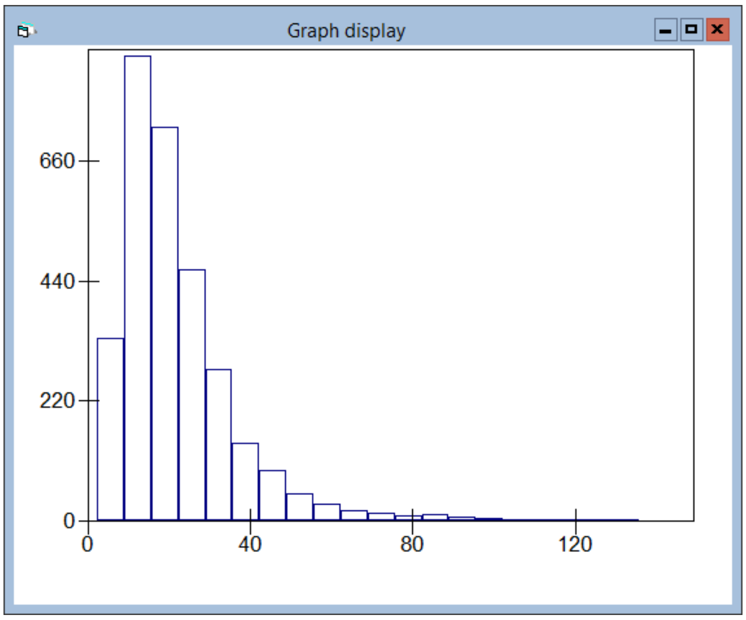
Name	Cn	n	missing	min	max	categorical	description
id	1	3022	0	1	3022	False	Unique office worker identifying code.
company	2	3022	0	1	141	False	Identifying code for company worked for over the last 12 months.
company2	3	3022	0	0	141	False	If worked for >1 company over the last 12 months, identifying code for sec
company3	4	3022	0	0	140	False	If worked for >2 company over the last 12 months, identifying code for thir
company4	5	3022	0	0	131	False	If worked for >3 company over the last 12 months, identifying code for fou
age	6	3022	0	16	65	False	Age of worker.
parttime	7	3022	0	0	1	False	Part or full-time (0=Fulltime, 1=Parttime).
sex	8	3022	0	0	1	False	Sex of worker (0=Male, 1=Female).
cons	9	3022	0	1	1	False	Constant (=1)
earnings	10	3022	0	2.4	135.6	False	Workers' earnings over the last financial year.
logearn	11	3022	0	0.8754688	4.909709	False	Workers' (natural) log-transformed earnings over the last financial year.
numjobs	12	3022	0	1	4	False	The number of companies worked for over the last 12 months.
weight1	13	3022	0	1.923077E-02	1	False	Proportion of time worked for employer listed in company.
weight2	14	3022	0	0	0.9807692	False	Proportion of time worked for employer listed in company2.
weight3	15	3022	0	0	0.8	False	Proportion of time worked for employer listed in company3.
weight4	16	3022	0	0	0.4	False	Proportion of time worked for employer listed in company4.
ew1	17	3022	0	0.25	1	False	Alternative (equal) weighting for company (1/numjobs).
ew2	18	3022	0	0	0.5	False	Alternative (equal) weighting for company2 (if numjobs >1 then 1/numjobs,
ew3	19	3022	0	0	0.334	False	Alternative (equal) weighting for company3 (if numjobs >2 then 1/numjobs,
ew4	20	3022	0	0	0.25	False	Alternative (equal) weighting for company4 (if numjobs >3 then 1/numjobs,
age-40	21	3022	0	-24	25	False	Age of worker, centered on 40 years.

Here we have as a response variable **earnings**, the amount (in thousands of pounds) that 3,022 workers earned in the last financial year. Most individuals worked for one company although some individuals worked for more, either because they work part time or because they changed job in the time period. For these individuals we have information on the names of all (up to 4) companies they worked for in columns labelled **company**, **company2**, **company3** and **company4**. There are 141 companies in the dataset and we also have the fraction of time the individuals worked for each company and this is stored in columns **weight1** to **weight4**. As predictors for salary we have the individual's age, sex, the number of jobs they worked on and whether they worked full or part-time.

We can firstly plot a histogram of the earnings:

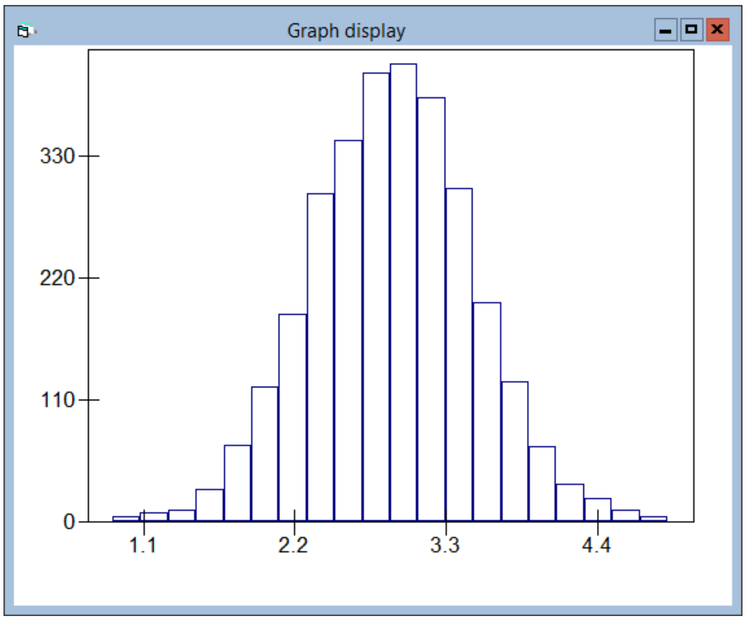
- Select **Customised Graph(s)** from the **Graphs** menu.
- Select **earnings** from the **y** pull down list.
- Select **histogram** from the **plot type** list.
- Click on the **Apply** button.

The graph will then appear as follows:



As is common with earnings data, the graph shows that the response is highly skewed to the right with the majority of people earning less than £40,000 while a few individuals earn over £100,000. Therefore it is probably better to consider a log transformation of the response and instead fit a Normal model to $\log_e$(response).

The column **logearn** was created via the command **CALC 'logearn' = LOGE ('earnings')** and is the (natural) logarithm of earnings. If we now plot a histogram of this variable instead (by changing the **y** column to **logearn**) we will see the following:



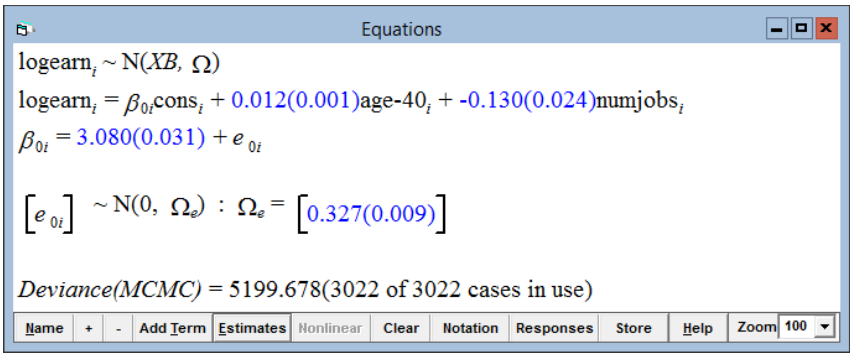
Here we see a much more Gaussian shape to the histogram suggesting that this will be a better variable to use as a response in a Normal response model.

16.3 Models for the earnings data

We will firstly consider fitting some simple regression models to the earnings data. The two predictors we will consider first are **age-40** and **numjobs** which represent the age of the workers (roughly centred) and the number of companies they have worked for in the last 12 months.

- Select **Equations** from the **Model** menu.
- Click on the **Notation** button and remove the tick for **multiple subscripts** and an i subscript will appear on the red y.
- Click on the **Done** button.
- Click on the red y_i and select **logearn** as the y variable.
- Select **2** as the number of classifications, **company** as **classification 2** and **id** as **classification 1**.
- Click on the red x_0 and select **cons** from the list.
- Select this variable as a **fixed parameter** and random at **level 1(id)**.
- Click on the **Add Term** button.
- Select **age-40** from the **variable** list and click on the **Done** button.
- Click on the **Add Term** button.
- Select **numjobs** from the **variable** list and click on the **Done** button.
- Click on the **Start** button.
- Change **Estimation method** to **MCMC**.
- Click on the **Start** button.

Upon finishing the 5,000 iterations we get the following estimates:



$\text{logearn}_i \sim \text{N}(XB, \Omega)$

$\text{logearn}_i = \beta_{0i}\text{cons}_i + 0.012(0.001)\text{age-40}_i + -0.130(0.024)\text{numjobs}_i$

$\beta_{0i} = 3.080(0.031) + e_{0i}$

$\left[e_{0i}\right] \sim \text{N}(0, \Omega_e) : \Omega_e = \left[0.327(0.009)\right]$

Deviance(MCMC) = 5199.678(3022 of 3022 cases in use)

So we see that older workers earn on average more whilst people who work for more companies in a year earn on average less. For example the average 40 year old with 1 job earns $e^{3.08-0.13}$=19.1k whilst the average 40 year old with 2 jobs earns $e^{3.08-0.26}$=16.8k. Note that due to the log transformation these estimates will be median earnings rather than means.

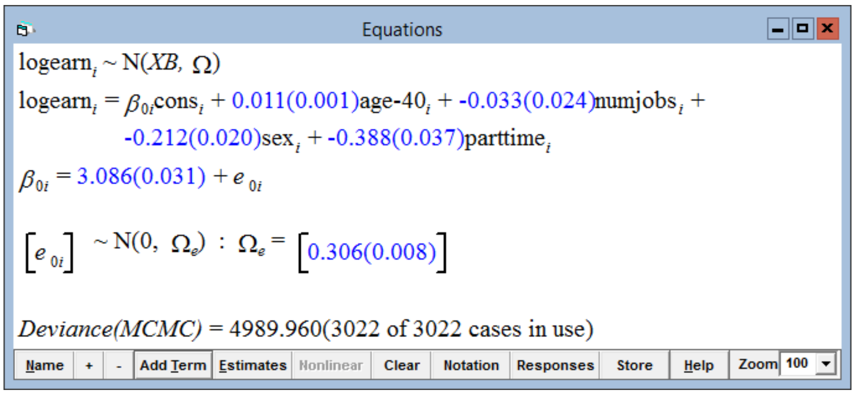
If we look at the DIC diagnostic we see that it confirms what we see from the significance of the predictors, that both predictors are important.

Dbar	D(thetabar)	pD	DIC	
5199.68	5195.62	4.06	5203.74	(age + numjobs)
5228.58	5225.56	3.03	5231.61	(age only)
5361.67	5359.66	2.01	5363.69	(neither)

We can now consider our two other predictor variables, gender and whether the person works full or part-time:

- Change **Estimation method** to **IGLS**.
- Click on the **Add Term** button.
- Select **sex** from the **variable** list and click on the **Done** button.
- Click on the **Add Term** button.
- Select **parttime** from the **variable** list and click on the **Done** button.
- Click on the **Start** button.
- Change **Estimation method** to **MCMC**.
- Click on the **Start** button.

Upon convergence we see the following estimates:



$$\text{logearn}_i \sim N(XB, \Omega)$$
$$\text{logearn}_i = \beta_{0i}\text{cons}_i + 0.011(0.001)\text{age-40}_i + -0.033(0.024)\text{numjobs}_i + -0.212(0.020)\text{sex}_i + -0.388(0.037)\text{parttime}_i$$
$$\beta_{0i} = 3.086(0.031) + e_{0i}$$
$$\left[e_{0i}\right] \sim N(0,\ \Omega_e) \ : \ \Omega_e = \left[0.306(0.008)\right]$$

Deviance(MCMC) = 4989.960(3022 of 3022 cases in use)

Here we see that both being female (**sex** = 1) and being a part time worker on average reduces your salary. It is interesting that putting these two variables into the model has reduced the negative effect of the number of jobs and it is now not significant. If we compare the DIC diagnostic for this model with the previous model, and a model with **numjobs** removed we see the following:

Dbar	D(thetabar)	pD	DIC	
4989.96	4983.98	5.98	4995.94	(this model)
5199.68	5195.62	4.06	5203.74	(age + numjobs only)
4990.83	4985.82	5.01	4995.84	(numjobs removed)

So it seems that the **numjobs** predictor is not important. If we now look at the correlations between the predictors:

- Select **Averages and Correlations** from the **Basic Statistics** menu.
- Select **Correlations** and the variables **parttime**, **sex** and **numjobs**.
- Select **Calculate**.

Correlations

	parttime	**sex**	**numjobs**
parttime	1.0000		
sex	0.0399	1.0000	
numjobs	0.2908	0.1014	1.0000

We can see here that there are (small) positive correlations between **numjobs** and both **sex** and **parttime** and so the significant effect for **numjobs** in the earlier model was a surrogate for the actual effects due to gender and part-time/full-time.

16.4 Fitting multiple membership models to the dataset

We can now consider accounting for the effects of the various companies on the earnings of the employees. If we look at the distribution of the number of jobs variable:

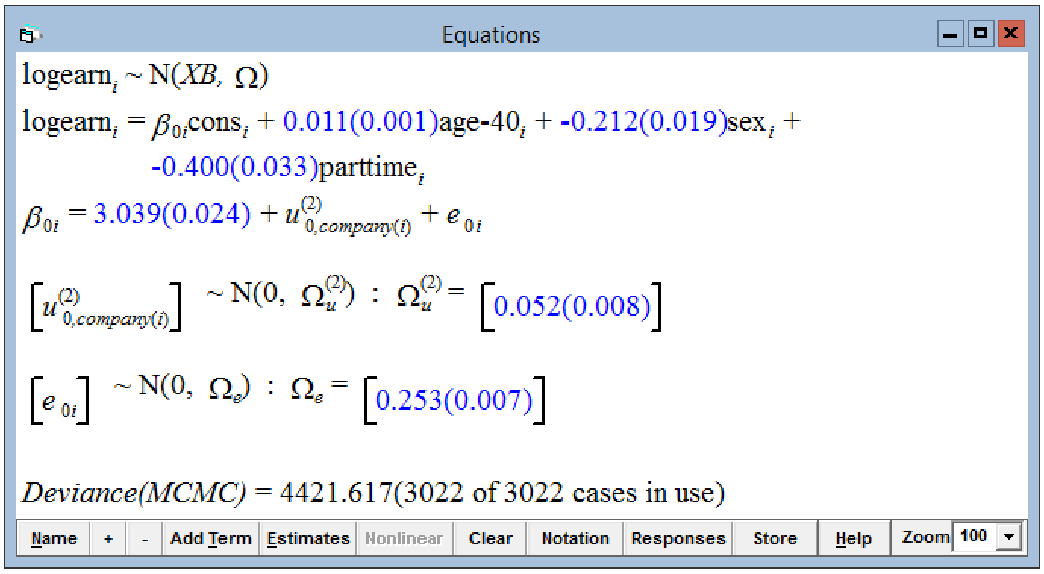
- Select **Tabulate** from the **Basic Statistics** window.
- Choose **numjobs** from the **Columns** pull down list.
- Click on the **Tabulate** button.

	1	2	3	4	TOTALS
N	2496	472	52	2	3022

We can see that in our dataset changing company is not a common phenomenon and so we could firstly fit a simple 2-level model that accounts for only one company per individual. This would be the type of model we would have to fit if, for example, we only collected the current employer for each individual. For our dataset we will consider just fitting the company that appears in the column **company** which will be the first in numerical order of the companies each individual works for.

- Change **Estimation method** to **IGLS**.
- Click on the β_2 (**numjobs**) and click on the **Delete Term** button.
- Click on the β_0 (**cons**) and tick the **company(2)** tick box.
- Click on the **Start** button.
- Change **Estimation method** to **MCMC**.
- Click on the **Start** button.

After running for 5,000 iterations the results are as follows:



$$\text{logearn}_i \sim \text{N}(XB, \Omega)$$
$$\text{logearn}_i = \beta_{0i}\text{cons}_i + 0.011(0.001)\text{age-40}_i + -0.212(0.019)\text{sex}_i + -0.400(0.033)\text{parttime}_i$$
$$\beta_{0i} = 3.039(0.024) + u^{(2)}_{0,company(i)} + e_{0i}$$
$$\left[u^{(2)}_{0,company(i)}\right] \sim \text{N}(0,\ \Omega^{(2)}_u) \ : \ \Omega^{(2)}_u = \left[0.052(0.008)\right]$$
$$\left[e_{0i}\right] \sim \text{N}(0,\ \Omega_e) \ : \ \Omega_e = \left[0.253(0.007)\right]$$

Deviance(MCMC) = 4421.617(3022 of 3022 cases in use)

So here we see that the first company of each employee explains 0.052/(0.052+0.253) = 17% of the remaining variation. If we look at the DIC diagnostic:

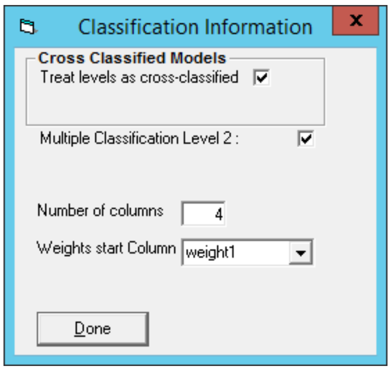
Dbar	D(thetabar)	pD	DIC	
4421.62	4311.72	109.90	4531.52	
4990.83	4985.82	5.01	4995.84	(no random effects)

Here we see that the random effects have reduced the DIC by over 400 and so are very important. There are also $110 - 5 = 105$ effective parameters for the 141 actual random effects so there are many important and distinct company effects.

If we now want to fit the multiple membership model we need to use the classifications window that we looked at in the last chapter:

- Change **Estimation method** to **IGLS** (to use same starting values).
- Click on the **Start** button.
- Change **Estimation method** to **MCMC**.
- Select **MCMC/Classifications** from the **Model** menu.
- Tick the **Multiple Classification Level 2** tickbox.
- Choose 4 for the **Number of columns**.
- Choose **weight1** as the **Weight start Column**.

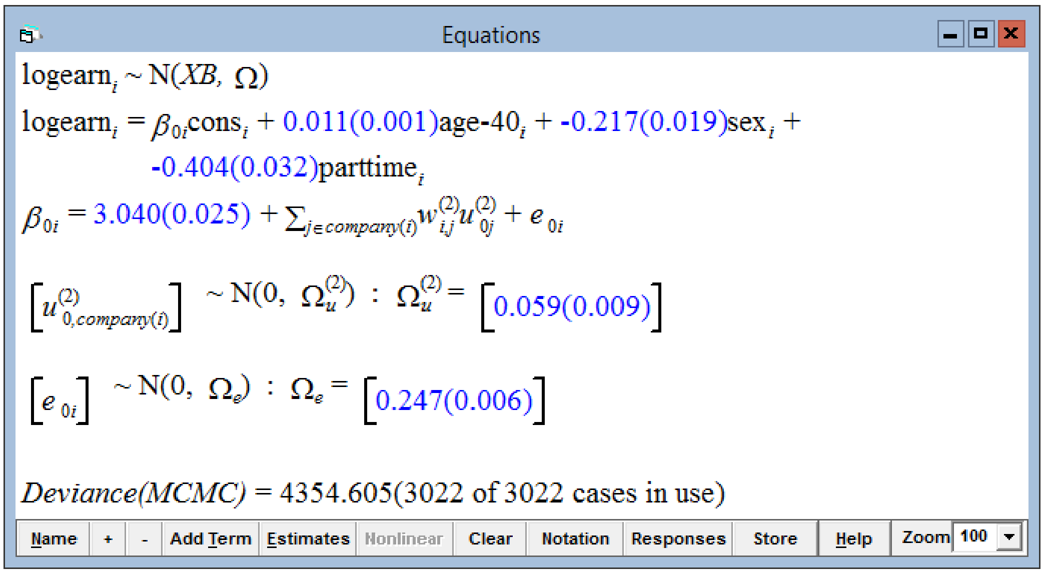
The window should look as follows:


Classification Information
Cross Classified Models
Treat levels as cross-classified ✓
Multiple Classification Level 2 : ✓
Number of columns 4
Weights start Column weight1

We now need to click on the **Done** button to apply the settings. Note that MLwiN assumes that the identifier columns are in sequential order, so in this example **company** (column **c2**) contains the first set of identifiers and the other three sets must be in the columns **c3–c5** respectively. The same is true for the weight columns with columns **c13–c16** named **weight1** to **weight4**. Note that if an observation is associated with less than 4 companies then both the identifier and weight for the extra companies should be set equal to zero. We have here used a system of filling in the end columns with zeroes although the order of the columns does not matter, as long as the ith weight column matches up with the ith identifier column.

- Click the **Start** button.

After running for 5,000 iterations we get the following estimates:



$$\text{logearn}_i \sim \text{N}(XB, \Omega)$$

$$\text{logearn}_i = \beta_{0i}\text{cons}_i + 0.011(0.001)\text{age-40}_i + -0.217(0.019)\text{sex}_i + -0.404(0.032)\text{parttime}_i$$

$$\beta_{0i} = 3.040(0.025) + \sum_{j \in company(i)} w^{(2)}_{i,j} u^{(2)}_{0j} + e_{0i}$$

$$\left[u^{(2)}_{0,company(i)}\right] \sim \text{N}(0, \Omega^{(2)}_u) : \Omega^{(2)}_u = \left[0.059(0.009)\right]$$

$$\left[e_{0i}\right] \sim \text{N}(0, \Omega_e) : \Omega_e = \left[0.247(0.006)\right]$$

Deviance(MCMC) = 4354.605(3022 of 3022 cases in use)

So accounting for all the companies explains 0.059/(0.059+0.247) = 19.3% of the variation. More importantly if we look at the DIC diagnostic we see that the DIC diagnostic has been reduced by over 60 and the number of effective parameters has increased, suggesting again many important and distinct company effects.

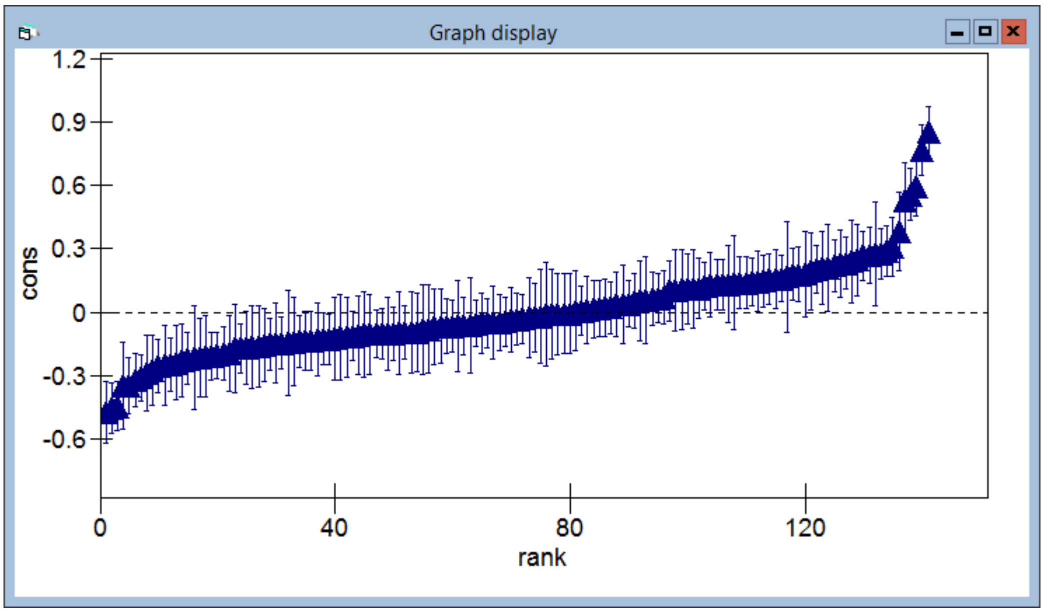
Dbar	D(thetabar)	pD	DIC	
4354.60	4240.64	113.97	4468.57	
4421.62	4311.72	109.90	4531.52	(no MM)

16.5 Residuals in multiple membership models

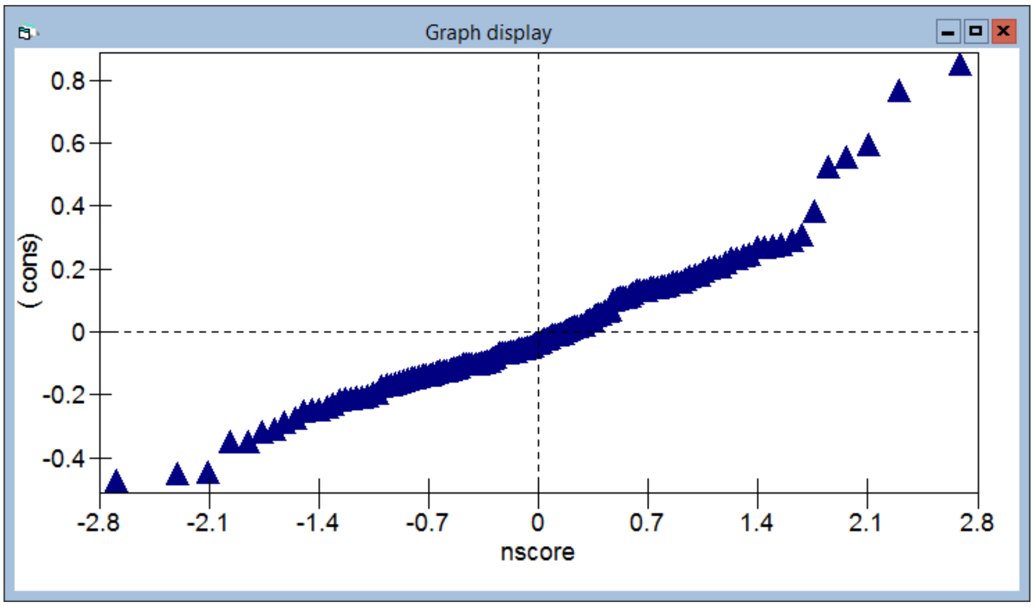
We can look at the individual company effects via the **Residuals** window.

- Select **Residuals** from the **Model** menu.
- Select **2:company** from the **level** pull down list.
- Change the **SD multiplier** from 1.0 to 1.4.
- Click on the **Calc** button.
- Click on the **Plots** tab.
- Select **residual +/- 1.4 sd x rank**.
- Click on the **Apply** button.

The residual graphs will then appear as shown below. Interestingly a couple of the companies with the highest residuals look like potential outliers.



Looking at a plot of the (raw) residuals against normalized scores also shows similar behaviour:



We can consider fitting separate terms for these two companies and treating these as fixed effects. Clicking on their residuals identifies them as companies 54 and 67. Normally we could use the graph window to create dummy variables for these two companies, but we cannot currently do this for multiple-membership models as MLwiN would currently only select individuals who have **company** = 54 (67) and ignore the other 3 potential companies. We can however currently create the two columns using the **Command Interface** window.

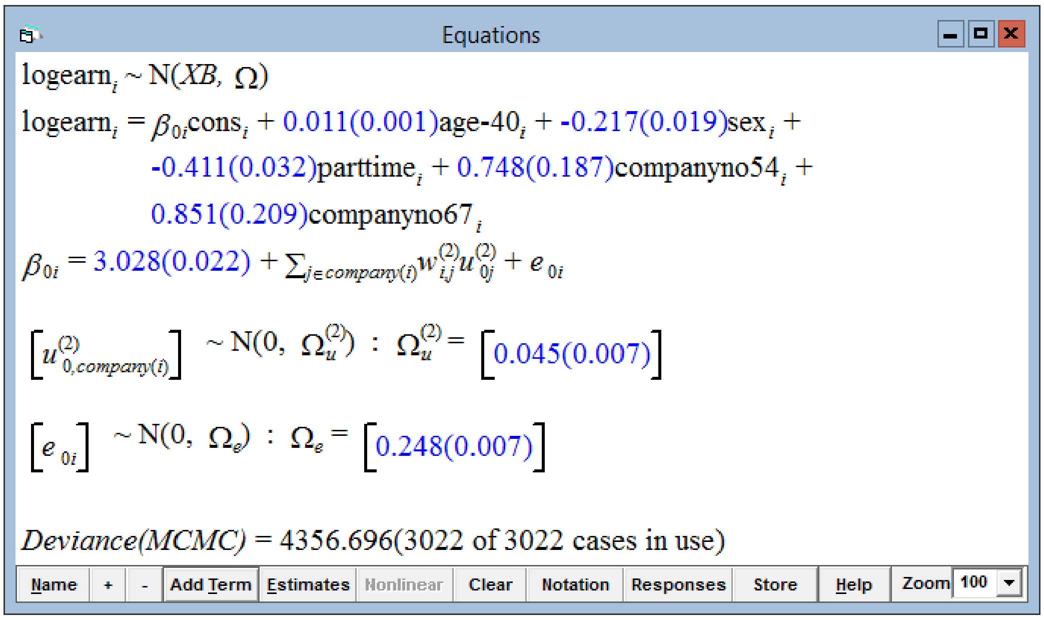
- Open the **Command Interface** window from the **Data Manipulation** menu.

- Type the following four commands:

```
▶ Calc c22 = ('company' == 54) + ('company2' == 54) +
  ('company3' == 54) + ('company4' == 54)
▶ name c22 'companyno54'
▶ Calc c23 = ('company' == 67) + ('company2' == 67) +
  ('company3' == 67) + ('company4' == 67)
▶ name c23 'companyno67'
```

- Change **Estimation method** to **IGLS**.
- Add the variables **companyno54** and **companyno67** as fixed effects to the model.
- Click on the **Start** button.
- Change **Estimation method** to **MCMC**.
- Click on the **Start** button.

Upon running for 5,000 iterations we get the following estimates:



$$\text{logearn}_i \sim \mathrm{N}(XB, \Omega)$$

$$\begin{aligned}\text{logearn}_i = {} & \beta_{0i}\text{cons}_i + 0.011(0.001)\text{age-40}_i + \text{-}0.217(0.019)\text{sex}_i + \\ & \text{-}0.411(0.032)\text{parttime}_i + 0.748(0.187)\text{companyno54}_i + \\ & 0.851(0.209)\text{companyno67}_i\end{aligned}$$

$$\beta_{0i} = 3.028(0.022) + \textstyle\sum_{j\in company(i)} w^{(2)}_{i,j} u^{(2)}_{0j} + e_{0i}$$

$$\left[u^{(2)}_{0,company(i)}\right] \sim \mathrm{N}(0,\ \Omega^{(2)}_u)\ :\ \Omega^{(2)}_u = \left[0.045(0.007)\right]$$

$$\left[e_{0i}\right] \sim \mathrm{N}(0,\ \Omega_e)\ :\ \Omega_e = \left[0.248(0.007)\right]$$

Deviance(MCMC) = 4356.696(3022 of 3022 cases in use)

Here we see that the two companies both have large positive coefficients and the company level variance has reduced from 0.059 to 0.045. If we look at the DIC diagnostic for the new model we see that it has been reduced by a further 4 suggesting this is a slightly improved model:

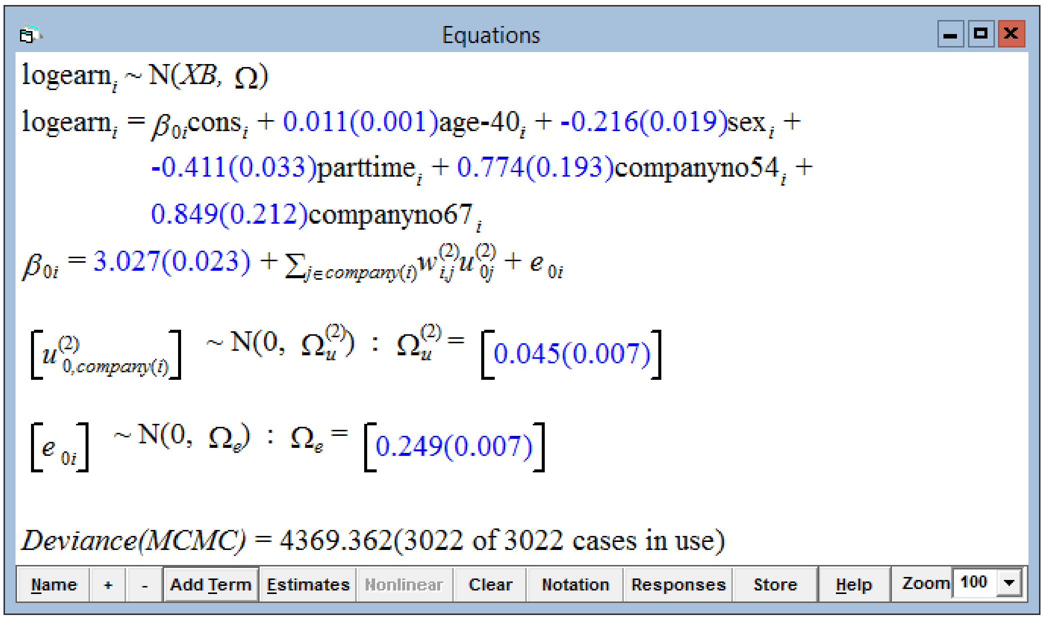
Dbar	D(thetabar)	pD	DIC	
4356.70	4248.63	108.07	4464.77	(with 2 additional fixed effects)
4354.60	4240.64	113.97	4468.57	(with all random effects)

16.6 Alternative weights for multiple membership models

We have so far used weights that are proportional to the time spent in each job and, as this is a simulated dataset, these were the weights that were actually used to generate the response variable. It is possible however to consider using other weightings, for example we could look at the effect of using equal weights so that each company you work for has an equal effect on your final salary.

- Change **Estimation method** to **IGLS**.
- Click on the **Start** button.
- Change **Estimation method** to **MCMC**.
- Select **MCMC/Classifications** from the **Model** menu.
- Change the **Weights** start column to **ew1**.
- Click on the **Done** button.
- Click on the **Start** button.

Upon running for 5,000 iterations we get the following estimates:



$$\text{logearn}_i \sim \text{N}(XB, \Omega)$$

$$\text{logearn}_i = \beta_{0i}\text{cons}_i + 0.011(0.001)\text{age-40}_i + -0.216(0.019)\text{sex}_i + -0.411(0.033)\text{parttime}_i + 0.774(0.193)\text{companyno54}_i + 0.849(0.212)\text{companyno67}_i$$

$$\beta_{0i} = 3.027(0.023) + \textstyle\sum_{j\in company(i)} w^{(2)}_{i,j} u^{(2)}_{0j} + e_{0i}$$

$$\left[u^{(2)}_{0,company(i)}\right] \sim \text{N}(0,\ \Omega^{(2)}_u) : \Omega^{(2)}_u = \left[0.045(0.007)\right]$$

$$\left[e_{0i}\right] \sim \text{N}(0,\ \Omega_e) : \Omega_e = \left[0.249(0.007)\right]$$

Deviance(MCMC) = 4369.362(3022 of 3022 cases in use)

The estimates are not greatly different from those we got when using the proportional weights earlier. This is probably because few respondents change job. However if we now look at the DIC diagnostic we see:

Dbar	D(thetabar)	pD	DIC	
4369.36	4261.93	107.43	4476.80	(equal weights)
4356.70	4248.63	108.07	4464.77	(proportional weights)

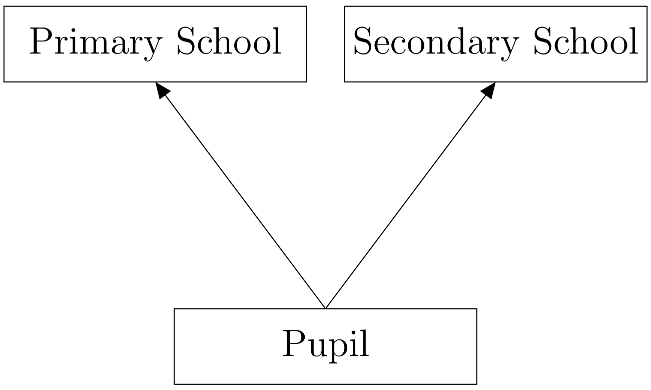
So we can see that the equal weights give a DIC that is 11 higher, which suggests that this is a worse model. This is not surprising given that we used the proportional weights to generate the response. However this shows that the DIC diagnostic is useful for choosing between possible weighting systems.

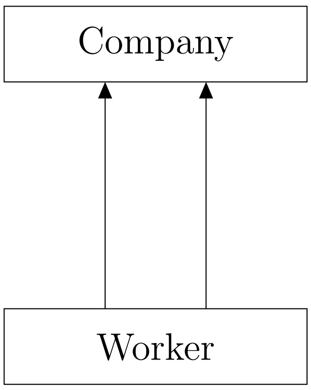
16.7 Multiple membership multiple classification (MMMC) models

In the past two chapters we have considered both cross-classified and multiple-membership models. Of course we can think of models that combine both these advancements in one model. For example in our above analysis we may also have information on what secondary school the workers attended and use this as a classification that is crossed with the multiple-membership classification for companies. We call such models multiple membership multiple classification (MMMC) models and Browne, Goldstein & Rasbash (2001*a*) give general MCMC algorithms for fitting such models. In their paper you will find two large examples from veterinary epidemiology and demography, which are too large to include in this manual. We will however consider the third example on Scottish lip cancer in the next chapter where amongst other models we will fit an MMMC model.

One other innovation in Browne et al. (2001*a*) is the '*classification diagram*'. When fitting MMMC models using MCMC estimation, knowledge of nesting relationships is not needed to implement the algorithms. For a simple summary of the model, however, it is useful to see the nesting relationships and so we advocate the use of a '*classification diagram*'. A classification diagram consists of boxes to represent each classification with arrows to represent nesting between two classifications, single arrows for single membership relationships and double arrows for multiple membership relationships.

Below are classification diagrams for the two examples in the last two chapters.





Chapter learning outcomes

- ⋆ What is meant by a multiple membership model.
- ⋆ How to transform and fit earnings data.
- ⋆ How to fit a multiple membership model in MLwiN.
- ⋆ How to look at residuals in a multiple membership model.
- ⋆ How to compare alternative weighting schemes.
- ⋆ What is meant by an MMMC model and a classification diagram.

Chapter 17

Modelling Spatial Data

In this chapter we will consider one particular dataset that has a spatial structure and discuss what models we can fit to such a dataset. The term *spatial data* will mean different things to different people. Here we will be as general as possible and consider collecting data at a variety of sites and also collecting some measure of the location of the sites. Of course we can analyse spatial data in the same way as any other dataset, for example in Chapter 11 we considered counts of melanoma mortality for different areas (locations) and simply fitted variance components models. However generally when we have spatial data we also wish to account for the effects of the locations.

How we adjust for location will depend on whether we have single observations at particular locations or estimates for contiguous areas. We can account for the effect of location in the first case by simply fitting functions of position, for example fitting polynomials in orthogonal directions or by considering point-process modelling. In the second case we are usually interested in fitting models that account for *spatial correlation*.

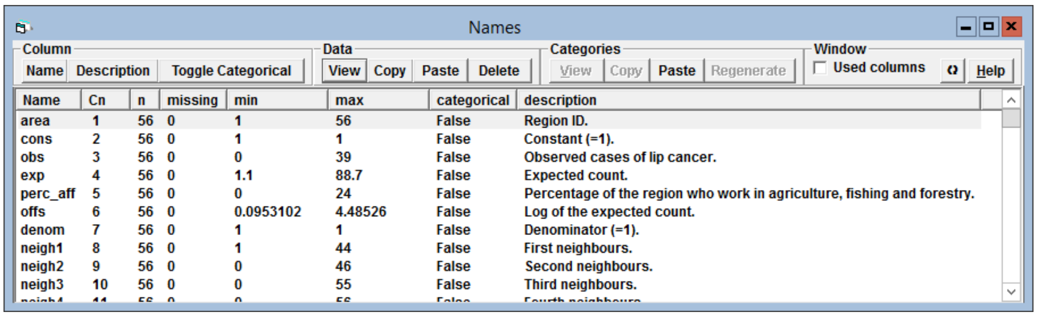
In earlier chapters we have considered fitting two level nested models and here we are assuming a correlation between individual level one units in the same level 2 unit. In a spatially correlated model we assume that the correlation between observations is some function of the spatial distance between them. In this chapter we will consider an epidemiological dataset of lip cancer incidence in Scotland.

17.1 Scottish lip cancer dataset

The dataset consists of observed counts of male lip cancer for the 56 regions of Scotland over the period 1975-1980 and was analysed in Clayton & Kaldor (1987). Research has focused on the effect of sun exposure on lip cancer deaths using the surrogate measure, percentage of the workforce working in outdoor occupations. The data are stored in the worksheet **lips1.ws**. First we will open up the worksheet and look at the variable names.

- Select **Open sample worksheet** from the **File** Menu.
- Select **lips1.ws** from the list and click on the **Open** button.

Here we see that for each of the 56 regions (**area**) we have observed cases of lip cancer (**obs**) and an expected count based on population size and mix (**exp**). As in the Melanoma example in Chapter 11 we have calculated the log of the expected count and this can be found in the column **offs**. The predictor of interest is **perc_aff**, which is the percentage of the region who work in agriculture, fishing and forestry and which ranges from 0% to 24%. We also have the neighbourhood structure for the regions, which is stored in columns **neigh1** to **neigh11**. Each region borders up to 11 other regions and the region numbers are stored in these columns.



Name	Cn	n	missing	min	max	categorical	description
area	1	56	0	1	56	False	Region ID.
cons	2	56	0	1	1	False	Constant (=1).
obs	3	56	0	0	39	False	Observed cases of lip cancer.
exp	4	56	0	1.1	88.7	False	Expected count.
perc_aff	5	56	0	0	24	False	Percentage of the region who work in agriculture, fishing and forestry.
offs	6	56	0	0.0953102	4.48526	False	Log of the expected count.
denom	7	56	0	1	1	False	Denominator (=1).
neigh1	8	56	0	1	44	False	First neighbours.
neigh2	9	56	0	0	46	False	Second neighbours.
neigh3	10	56	0	0	55	False	Third neighbours.

17.2 Fixed effects models

We will firstly fit a simple constant risk model, which assumes that there is an underlying risk of getting lip cancer which is constant across the whole population and any deviations from this risk in particular regions are simply random Poisson variation. Note that in this chapter we will again use the notation introduced in Browne, Goldstein & Rasbash (2001*a*). We can set up this model as follows:

- Open the **Equations** window from the **Model** Menu.
- Click on the **Notation** button and remove the tick for **multiple subscripts** and an i subscript will appear on the red y.
- Click on the **Done** button.
- Click on the red y_i.
- Select **obs** as the y variable.
- Select **3** as the number of classifications.

- Select **neigh1** as **classification 3**.
- Select **area** as **classification 2**.
- Select **area** as **classification 1**.
- Click on the **Done** button.
- Click on the **N** and select **Poisson** from the list.
- Click on the **Done** button.
- Click on the $\log(\pi_i)$ and select **offs** from the list.
- Click on the **Done** button.

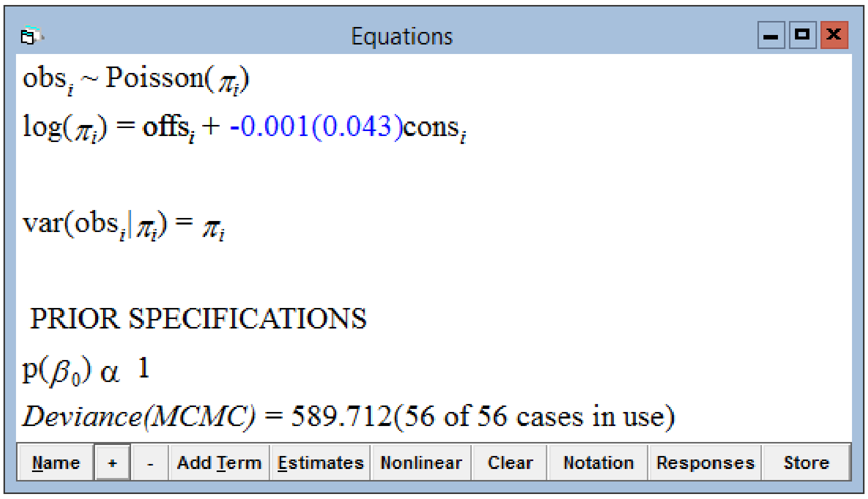
Note that here we have set up the response and its Poisson distribution. We have also defined the (potential) full structure of the dataset, which is not important for this first model but will be used later. Both classifications 1 and 2 are defined as **area** because there is only 1 observation per area and so classification 2 will be used to account for over-dispersion in later models. We have also defined the offset parameter to be **offs**. We now need to set up the predictor variables. We will start with just an intercept term.

- Click on the red x_0 and select **cons**
- Click on the **Done** button.

As always we will run the model using IGLS first for starting values.

- Click on the **Start** button.
- Change **Estimation mode** to **MCMC**.
- Click on the **Start** button.

When the 5,000 iterations are finished we get the following estimates:



$\text{obs}_i \sim \text{Poisson}(\pi_i)$

$\log(\pi_i) = \text{offs}_i + -0.001(0.043)\text{cons}_i$

$\text{var}(\text{obs}_i|\pi_i) = \pi_i$

PRIOR SPECIFICATIONS

$p(\beta_0) \propto 1$

Deviance(MCMC) = 589.712(56 of 56 cases in use)

We see that the estimate of the fixed effect is approximately zero. All this is saying is that on average the observed count equals the expected count, which is to be expected. This is because the offset is the expected count. If instead the offset had been the number of people in the population at risk then the intercept would equal the observed log-rate. More importantly we get an estimate of the Poisson deviance for the model and we can get the DIC diagnostic for this model from the **Model** menu:

Dbar	D(thetabar)	pD	DIC
589.71	588.70	1.01	590.72

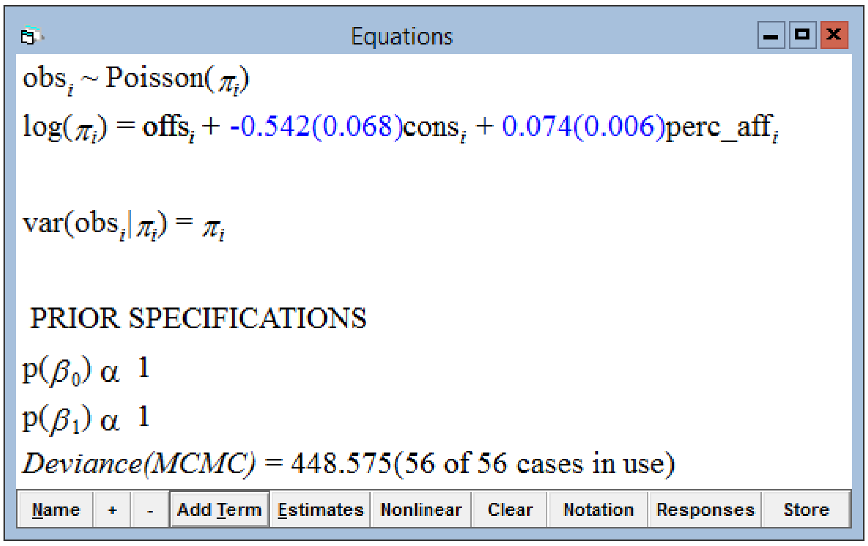
We have one covariate of interest, **perc_aff** which is the percentage of the population working in outdoor activities and we can add this to the model as follows:

- Change **Estimation mode** to **IGLS**.
- Click on the **Add term** button.
- Select **perc_aff** from the **variable** list and click on the **Done** button.

We can now run this model:

- Click on the **Start** button.
- Change **Estimation mode** to **MCMC**.
- Click on the **Start** button.

When the estimation finishes we get the following estimates:



$\text{obs}_i \sim \text{Poisson}(\pi_i)$

$\log(\pi_i) = \text{offs}_i + -0.542(0.068)\text{cons}_i + 0.074(0.006)\text{perc\_aff}_i$

$\text{var}(\text{obs}_i|\pi_i) = \pi_i$

PRIOR SPECIFICATIONS

$p(\beta_0) \propto 1$

$p(\beta_1) \propto 1$

Deviance(MCMC) = 448.575(56 of 56 cases in use)

Here we see that the percentage of workers in agriculture, forestry and fishing is a significant predictor of lip cancer incidence with a positive association. This model fits a lot better as the DIC diagnostic shows:

Dbar	D(thetabar)	pD	DIC	
448.58	446.60	1.98	450.55	(with predictor)
589.71	588.70	1.01	590.72	(without predictor)

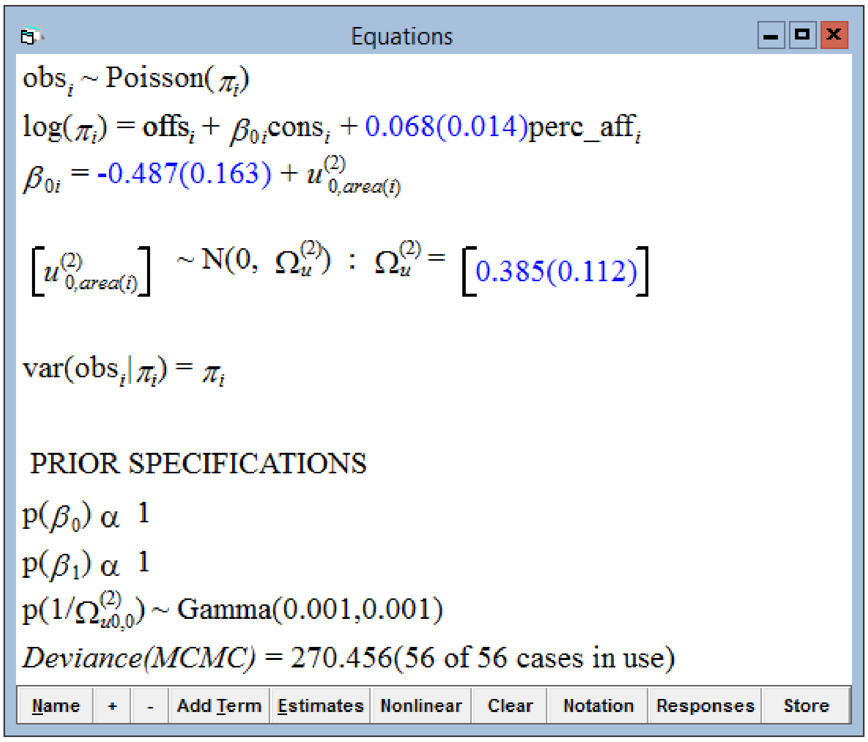
17.3 Random effects models

As with the melanoma example in Chapter 11 the next step is to consider fitting random effects to explain the remaining variation in the data. The simplest model here is to fit a variance components model where each region is given an (exchangeable) random effect, and these random effects are assumed to have a Normal distribution. This will now mean that for our 56 data points we are fitting 58 parameters (2 fixed effects and 56 random effects) but this is allowed as the random effects are linked by their Normal distribution assumption, so in terms of effective parameters we are fitting less than 58 parameters.

As we also saw in Chapter 11, Poisson models often produce highly autocorrelated chains and so for the rest of the models in this chapter we will increase the lengths of the stored MCMC chains to 50,000. We will consider methods of improving mixing of chains for Poisson models in later chapters. To fit the variance components model we need to do the following:

- Change **Estimation mode** to **IGLS**.
- Click on the x_0 (**cons**) and select the **area(2)** tick box .
- Click on the **Done** button.
- Click on the **Start** button.
- Change **Estimation mode** to **MCMC**.
- Change the **monitoring chain length** to 50,000 on the **Estimation Control** window.
- Change the **refresh rate** to 500 on the **Estimation Control** window.
- Click on the **Start** button.

Upon completion of the 50,000 iterations we will get the following results:



$\text{obs}_i \sim \text{Poisson}(\pi_i)$

$\log(\pi_i) = \text{offs}_i + \beta_{0i}\text{cons}_i + 0.068(0.014)\text{perc\_aff}_i$

$\beta_{0i} = -0.487(0.163) + u^{(2)}_{0,area(i)}$

$\left[u^{(2)}_{0,area(i)}\right] \sim \text{N}(0,\ \Omega^{(2)}_u)\ :\ \Omega^{(2)}_u = \left[0.385(0.112)\right]$

$\text{var}(\text{obs}_i|\pi_i) = \pi_i$

PRIOR SPECIFICATIONS

$\text{p}(\beta_0) \propto 1$

$\text{p}(\beta_1) \propto 1$

$\text{p}(1/\Omega^{(2)}_{u0,0}) \sim \text{Gamma}(0.001,0.001)$

Deviance(MCMC) = 270.456(56 of 56 cases in use)

Here we see the variance of the random effects appears to be quite large (0.384) and the deviance has been reduced considerably. To compare the models we can look at the DIC diagnostic:

Dbar	D(thetabar)	pD	DIC	
270.46	230.67	39.79	310.24	(with random effects)
448.58	446.60	1.98	450.55	(with no random effects)

The DIC diagnostic is reduced by 140 suggesting a substantially improved model, and we can also see that the 58 parameters translate to only 40 effective parameters.

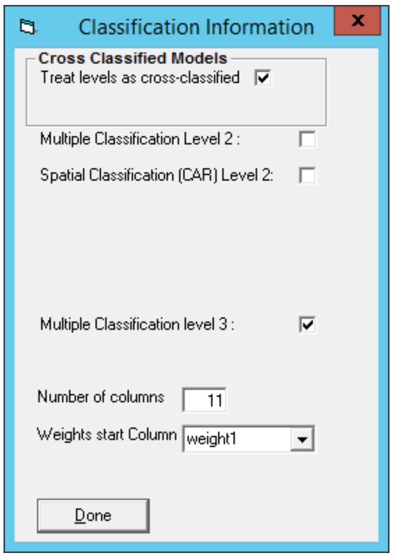
17.4 A spatial multiple-membership (MM) model

We have so far not taken account of the spatial relationships in the dataset apart from assuming the areas are separate entities. In the lip cancer data the only spatial information we have is a list of which regions border each other. In other words we have, for each region, its nearest neighbours. We will therefore consider two sets of random effects in our model. Firstly we will consider the exchangeable area random effects that we have already fitted and then a multiple membership set of random effects for the neighbours of each region. This means that the rate of lip cancer in each region is affected by both the region itself and its nearest neighbours.

This model can be set up as follows:

- Change **Estimation mode** to **IGLS**.
- Click on the x_0 (**cons**) and select the **neigh1(3)** tick box.
- Click on the **Done** button.
- Click on the **Start** button (Click **yes** on any numeric error messages).
- Change **Estimation mode** to **MCMC**.
- Select **MCMC/Classifications** from the **Model** menu.
- Tick the **Multiple Classification level 3** tick box on the **Classification information** window.
- Change the **Number of Columns** to 11.
- Change the **Weights start column** to **weight1**.

The **Classification information** window should now look as follows:



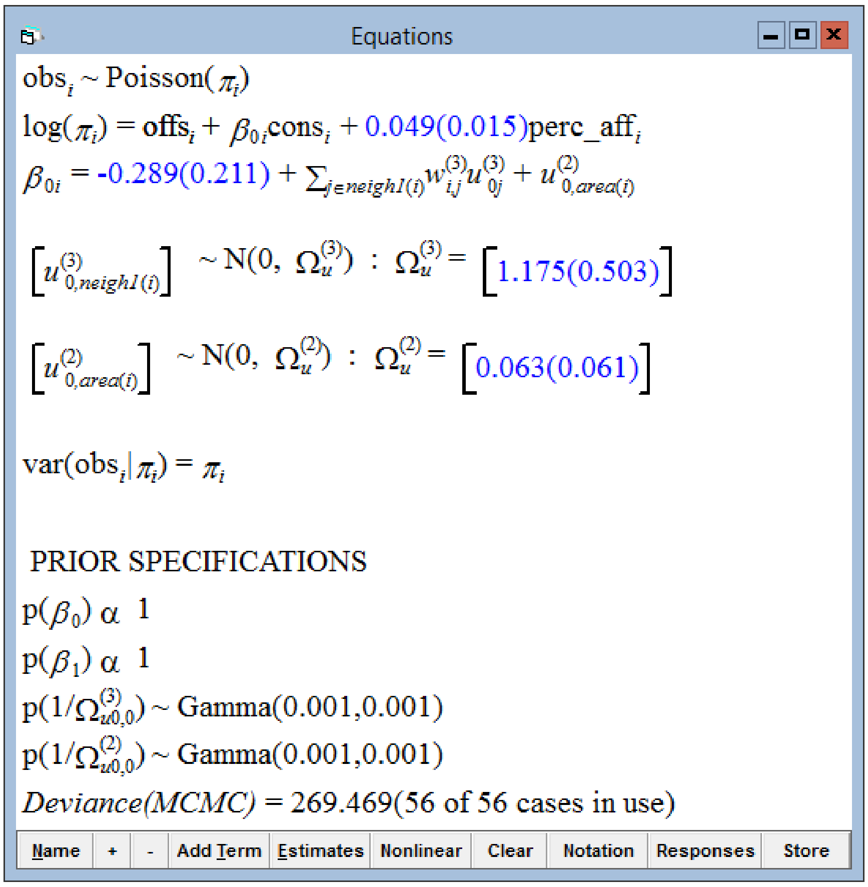
Treat levels as cross-classified ✓
Multiple Classification Level 2 :
Spatial Classification (CAR) Level 2:
Multiple Classification level 3 : ✓
Number of columns 11
Weights start Column weight1

The weight columns contain equal weights for each neighbouring region that sum to 1. This means that a region with four neighbours will have weights of 0.25 for the four neighbouring regions (stored in columns **weight1** to **weight4** with zeroes in the columns **weight5** to **weight11**). As in the last chapter the neighbouring region identifiers are stored in sequential columns starting from **neigh1** and there are at most 11 nearest neighbours for each region.

To run the model we now need to do the following:

- Click on the **Done** buttons on the **Classification** window.
- Click on the **Start** button.

After the 50,000 iterations have run the results will be as follows:



$\text{obs}_i \sim \text{Poisson}(\pi_i)$

$\log(\pi_i) = \text{offs}_i + \beta_{0i}\text{cons}_i + 0.049(0.015)\text{perc\_aff}_i$

$\beta_{0i} = -0.289(0.211) + \sum_{j \in neigh1(i)} w^{(3)}_{i,j} u^{(3)}_{0j} + u^{(2)}_{0,area(i)}$

$\left[u^{(3)}_{0,neigh1(i)}\right] \sim \text{N}(0,\ \Omega^{(3)}_u)\ :\ \Omega^{(3)}_u = \left[1.175(0.503)\right]$

$\left[u^{(2)}_{0,area(i)}\right] \sim \text{N}(0,\ \Omega^{(2)}_u)\ :\ \Omega^{(2)}_u = \left[0.063(0.061)\right]$

$\text{var}(\text{obs}_i|\pi_i) = \pi_i$

PRIOR SPECIFICATIONS

$p(\beta_0) \propto 1$

$p(\beta_1) \propto 1$

$p(1/\Omega^{(3)}_{u0,0}) \sim \text{Gamma}(0.001,0.001)$

$p(1/\Omega^{(2)}_{u0,0}) \sim \text{Gamma}(0.001,0.001)$

Deviance(MCMC) = 269.469(56 of 56 cases in use)

The deviance for this model is reduced slightly but we have added another set of 56 random effects making a total of 114 parameters for 56 data points! Interestingly, if we look at the DIC diagnostic, adding this second set of parameters actually reduces the effective number of parameters:

Dbar	D(thetabar)	pD	DIC	
269.47	236.20	33.27	302.74	(exchangeable + MM)
270.46	230.67	39.79	310.24	(exchangeable random effects only)

The DIC diagnostic shows that this model is a large improvement on the last model. To explain the reduction in effective parameters, comparing the last models it can be seen that the variance associated with the exchangeable random effects is greatly reduced by the addition of the second set of effects. As the effective number of parameters for these random effects lies somewhere between 0 and 56 and a variance of 0 would be equivalent to an effective number of parameters equal to zero, then a reduction in the variance implies a reduced effective number of parameters. Obviously this has to be balanced by the additional parameters due to the second set of random effects and so here the reduction must be greater than the effective number of parameters introduced by the additional set of random effects.

Langford et al. (1999) described the use of multiple membership models in spatial applications using pseudo-likelihood methods. They also extended the model to include a correlation between the two sets of random effects. This extension could also be done using MCMC but is currently not available in MLwiN.

17.5 Other spatial models

There are more standard ways of fitting spatial models using MCMC to Poisson data and these are based on the conditional autoregressive (CAR) prior (Besag et al., 1992) that was originally used in image analysis. These priors were used on the Scottish lip cancer dataset originally in Breslow & Clayton (1993).

The CAR prior is a spatial smoothing prior and CAR models differ from the multiple membership (MM) model we have just looked at because the individual random effects are not independent. We can write a CAR model as follows:

$$\begin{aligned}
\mathbf{obs}_i &\sim \mathbf{Poisson}(\pi_i) \\
\log_e(\pi_i) &= \log_e(\mathbf{exp}_i) + X_i\beta_2 + u^{(2)}_{\mathbf{area}[i]} \\
u^{(2)}_{\mathbf{area}[i]} &\sim \mathrm{N}(\bar{u}^{(2)}_{\mathbf{area}[i]}, \sigma^2_{u(2)}/r_{\mathbf{area}[i]}) \\
\text{where } \bar{u}^{(2)}_{\mathbf{area}[i]} &= \sum_{j \in \mathbf{neighbour}(\mathbf{area}[i])} w^{(2)}_{\mathbf{area}[i],j} u^{(2)}_j / r_{\mathbf{area}[i]}
\end{aligned}$$

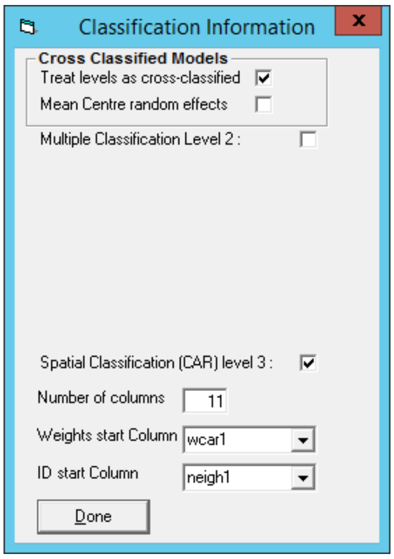
This model has only one set of random effects, although it is also possible to fit a model with an additional set of exchangeable random effects as we will show later. The difference between the CAR model and the MM model is that whilst the MM model has $r_{\mathbf{area}[i]}$ random effects for each observation, where $r_{\mathbf{area}[i]}$ is the number of neighbours for region i, the CAR model has one random effect for each observation. These random effects have as expected value the average of the surrounding random effects. Note to make the CAR model identifiable we either need to constrain the random effects to sum to 0 or remove the intercept from the model (as shown above).

17.6 Fitting a CAR model in MLwiN

CAR models are not standard multilevel models and so have only been added to MLwiN for the MCMC methods and haven't been extensively tested. We therefore suggest you compare the results produced for any CAR model with the results from the WinBUGS package. To set up the above CAR model we need to do the following:

- Change **Estimation mode** to **IGLS**.
- Click on the y_i (**obs**) and change **classification 3** to **area**.
- Click on the **Done** button.
- Click on the x_0 (**cons**) and remove the ticks for **fixed effect** and **area(2)**.
- Click on the **Done** button.
- Click on the **Start** button.
- Change **Estimation mode** to **MCMC**.
- Select **MCMC/Classifications** from the **Model** menu.
- Remove the tick for **Multiple Classification level 3**.
- Tick the **Spatial Classification (CAR) level 3** tick box.
- Select 11 for the **number of columns**.
- Select **wcar1** for the **Weights start Column**.
- Select **neigh1** for the **ID start Column**.

The Classifications window should now look as follows:



Classification Information

Cross Classified Models
Treat levels as cross-classified ✓
Mean Centre random effects ☐

Multiple Classification Level 2 : ☐

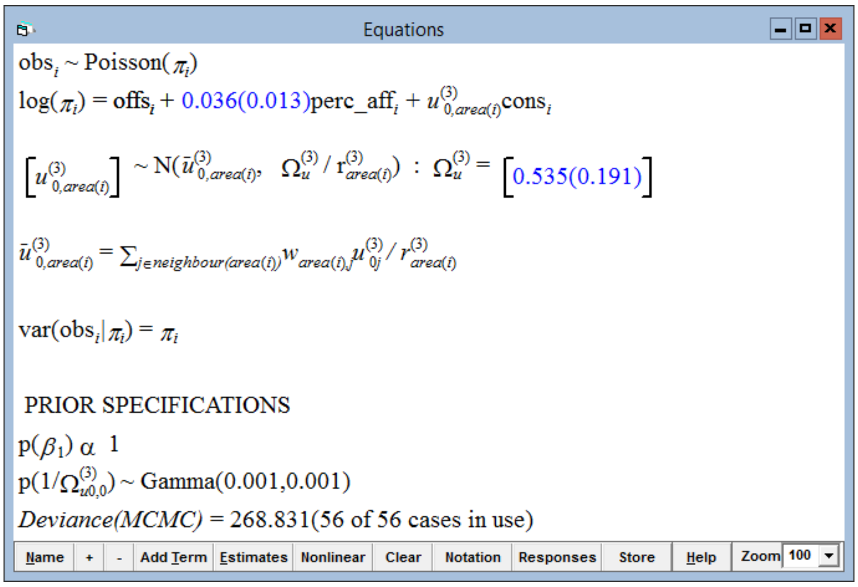
Spatial Classification (CAR) level 3 : ✓
Number of columns 11
Weights start Column wcar1
ID start Column neigh1

Note that previous versions of MLwiN only allowed one set of CAR residuals. In this version of MLwiN you can mean centre the random effects and in this case more than one set of CAR residuals are permissable. Also for the CAR model (unlike the MM model) we must give the column for the first set of neighbours on this screen as the classification given in the Equations window now gives the actual area codes for each observation. This is because for CAR residuals we need to know both the neighbouring regions and the actual region, whereas MM residuals are not always used for spatial models and hence do not always have an associated actual classification (see example in Chapter 16). Also the CAR procedure typically uses weights of 1 for all observations, as these weights will then be divided by the number of neighbours in the model (n_i).

We now need to run this model:

- Click on the **Done** buttons on the **Classification information** window.
- Click on the **Start** button.

After the 50,000 updates have run we get the following results:



$\text{obs}_i \sim \text{Poisson}(\pi_i)$

$\log(\pi_i) = \text{offs}_i + 0.036(0.013)\text{perc\_aff}_i + u^{(3)}_{0,area(i)}\text{cons}_i$

$\left[u^{(3)}_{0,area(i)}\right] \sim \text{N}(\bar{u}^{(3)}_{0,area(i)},\ \Omega^{(3)}_u / r^{(3)}_{area(i)}) \ :\ \Omega^{(3)}_u = \left[0.535(0.191)\right]$

$\bar{u}^{(3)}_{0,area(i)} = \sum_{j \in neighbour(area(i))} w_{area(i),j} u^{(3)}_{0j} / r^{(3)}_{area(i)}$

$\text{var}(\text{obs}_i | \pi_i) = \pi_i$

PRIOR SPECIFICATIONS

$p(\beta_1) \propto 1$

$p(1/\Omega^{(3)}_{u0,0}) \sim \text{Gamma}(0.001,0.001)$

Deviance(MCMC) = 268.831(56 of 56 cases in use)

If we compare this model to the multiple membership model via the DIC diagnostic we get the following estimates:

Dbar	D(thetabar)	pD	DIC	
268.83	240.47	28.36	297.19	(CAR model)
269.47	236.20	33.27	302.74	(MM model)

Here we can see that the CAR model shows an improvement of 5 over the multiple-membership model. Browne, Goldstein & Rasbash (2001*a*) did some other comparisons between the CAR and MM models on the lip cancer dataset and also found a slight improvement with the CAR model.

Here we see that we get an estimate for the effect of the outdoor activity predictor and the variance of the CAR residuals but no estimate for the intercept. We can still get a point estimate for the intercept by calculating the residuals and finding their average.

- Select **Residuals** from the **Model** menu.
- Select **3:area** for the level indicator.
- Click on the **Calc** button (ignore any error message here).
- Select **Averages and Correlations** from the **Basic Statistics** menu.
- Select column **c300** from the pull down list and click on the **Calculate** button.

The following results will appear:

	N	Missing	Mean	s.d.
c300	56	0	-0.21058	0.58506

So we see the intercept estimate is -0.211. (Note the s.d. here does not correspond to the standard error for this parameter).

We can now consider fitting this model in WinBUGS. To save the worksheet in WinBUGS format we need to do the following:

- Select **MCMC/WinBugs Options** from the **Model** menu.
- Select the **WinBugs 1.4** button.
- Click on the **Save Current Model in BUGS format** button.
- Change **Save as type** to **.bug files (*.bug)**
- Enter the filename **car.bug** and click on the **Save** button.

If we now start up WinBUGS and read in the file **car.bug** as a text file (from the directory it was saved) we will see the following (note you will need to change the Files of type box to **All files (*.*)** to see the file **car.bug**) :

```
#WINBUGS 1.4 code generated from MLwiN program

#----MODEL Definition----------------

model
{
# Level 1 definition
for(i in 1:N) {
obs[i] ~ dpois(mu[i])
log(mu[i]) <- offs[i] + beta[1] * perc_aff[i]
+ carmean + u3[area[i]] * cons[i]
}
# Higher level definitions
u3[1:n3] ~ car.normal(adj[],weights[],num[],tau.u3)
# Priors for fixed effects
beta[1] ∼ dflat()
carmean ∼ dflat()
# Priors for random terms
tau.u3 ∼ dgamma(0.001000,0.001000)
sigma2.u3 <- 1/tau.u3
}
```

Here we see the model definition has the special function **car.normal** that defines the CAR residuals and accounts for the weights and neighbouring regions. The BUGS code also monitors the intercept, which is called **carmean** in the above code.

Note that the BUGS examples volume II (Spiegelhalter et al., 2000*b*) also contains this lip cancer example dataset.

We can now set up the model and load in the data and initial values via the **Specification** window available from the **Model** menu (see Chapter 7 for details). We will monitor the parameters **beta**, **carmean** and **sigma2.u3** by using the **Samples** window available from the **Inference** menu. Here you should also change the **beg** box to 501 to allow a burnin of 500 iterations.

Now select the **Update** window from the **Model** menu and modify the **number of updates** to 50,500 (500 for the burnin) and click on the **Update** button.

Once the updating has finished we can use the **Samples** window to get estimates. If we type * in the **node** box and click on **stats** we get the summary statistics for all monitored parameters as shown below:

```
node       mean     sd      MC error  2.5%      median   97.5%
beta[1]    0.03494  0.0131  2.802E-4  0.008248  0.03521  0.05976
carmean    -0.2036  0.1201  0.00249   -0.4365   -0.2037  0.03364
sigma2.u3  0.5671   0.2001  0.002295  0.2702    0.5359   1.042
```

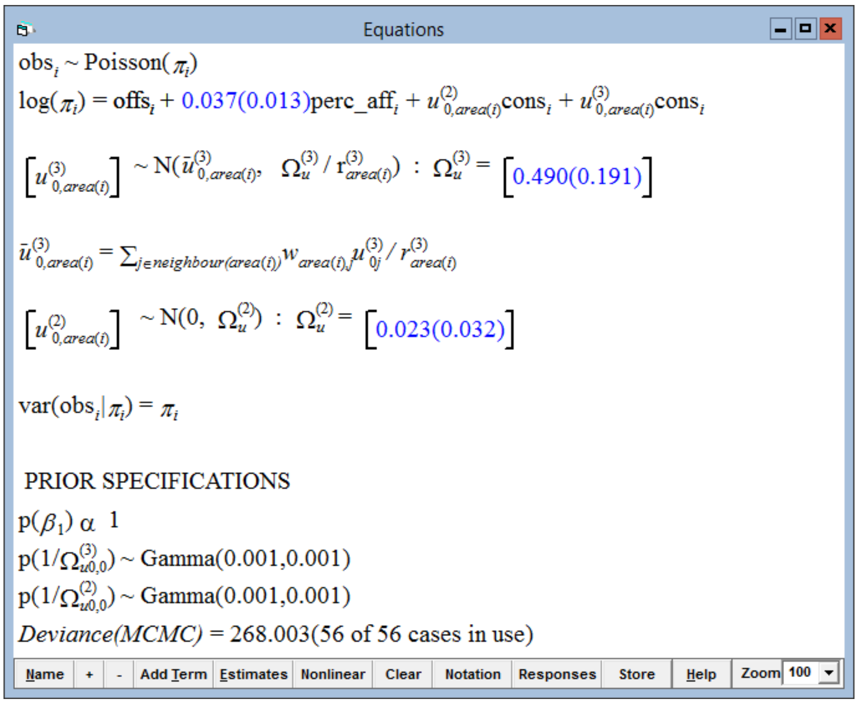
Here we see a reasonable agreement with the MLwiN estimates and we additionally get the standard error for the intercept term.

17.7 Including exchangeable random effects

As was mentioned earlier we can extend our model to include exchangeable random effects as we did for the multiple membership model. This model is often described as a *convolution* model (Besag et al., 1992). To set up the model in MLwiN we need to do the following:

- Change **Estimation mode** to **IGLS**.
- Click on x_0 (**cons**) and tick the **area(2)** tickbox and then the **Done** button.
- Click on the **Start** button (Click **yes** on any numeric error messages).
- Change **Estimation mode** to **MCMC**.
- Click on the **Start** button.

Upon completion of the 50,000 iterations the estimates are as follows:



$\text{obs}_i \sim \text{Poisson}(\pi_i)$

$\log(\pi_i) = \text{offs}_i + 0.037(0.013)\text{perc\_aff}_i + u^{(2)}_{0,area(i)}\text{cons}_i + u^{(3)}_{0,area(i)}\text{cons}_i$

$\left[u^{(3)}_{0,area(i)}\right] \sim \text{N}(\bar{u}^{(3)}_{0,area(i)},\ \Omega^{(3)}_u / r^{(3)}_{area(i)}) \ :\ \Omega^{(3)}_u = \left[0.490(0.191)\right]$

$\bar{u}^{(3)}_{0,area(i)} = \sum_{j \in neighbour(area(i))} w_{area(i),j} u^{(3)}_{0j} / r^{(3)}_{area(i)}$

$\left[u^{(2)}_{0,area(i)}\right] \sim \text{N}(0,\ \Omega^{(2)}_u) \ :\ \Omega^{(2)}_u = \left[0.023(0.032)\right]$

$\text{var}(\text{obs}_i|\pi_i) = \pi_i$

PRIOR SPECIFICATIONS

$p(\beta_1)\ \alpha\ 1$

$p(1/\Omega^{(3)}_{u0,0}) \sim \text{Gamma}(0.001,0.001)$

$p(1/\Omega^{(2)}_{u0,0}) \sim \text{Gamma}(0.001,0.001)$

Deviance(MCMC) = 268.003(56 of 56 cases in use)

The deviance has been reduced slightly but if we look at the DIC diagnostic we see that this model is slightly worse due to its added complexity, although there is not much difference in DIC value.

Dbar	D(thetabar)	pD	DIC	
268.00	238.38	29.63	297.63	(Convolution model)
268.83	240.47	28.36	297.19	(CAR model)

17.8 Further reading on spatial modelling

There is a very large literature in spatial statistics and here we have only mentioned a few of the possible spatial models. Mollie (1996) gives a good overview of spatial modelling of epidemiology data from a Bayesian perspective. Lawson et al. (1999) and Elliott et al. (2000) are also good books on the subject of spatial epidemiology. WinBUGS has an add-on package called GeoBUGS (Thomas et al., 2000), which includes further features in spatial modelling including facilities to read in data from maps. Lawson et al. (2003) gives further details of fitting spatial models in both MLwiN and WinBUGS.

Both WinBUGS and MLwiN use univariate updating routines for Poisson models and CAR models are one class of models where the block-updating Metropolis methods of Rue (2001) can be used efficiently (see Knorr-Held & Rue, 2002). We also consider other methods for improving mixing in Poisson models in chapters 23 and 25.

Chapter learning outcomes

- ⋆ How to fit various spatial models to datasets.
- ⋆ How to extend the multiple membership model to spatial applications.
- ⋆ What a CAR model and a convolution model are.
- ⋆ How to fit CAR models in both MLwiN and WinBUGS.
- ⋆ How to compare various spatial models using the DIC diagnostic.

Chapter 18

Multivariate Normal Response Models and Missing Data

We have so far concentrated on problems where we have one distinct 'response' variable that we are interested in and any other variables in our dataset are treated as predictor variables for our response. Whether a variable is chosen as a predictor or a response in an analysis generally depends on the research questions that we have formed. Sometimes factors such as collection time will govern our choice, for example in the tutorial dataset it would not make much sense to treat the exam scores at 16 as predictors for the London reading test scores which were taken five years earlier.

We may find however that our research questions result in us identifying several variables as responses. We could then fit models to each response variable in turn using the techniques we have so far discussed. This will result in separate analyses for each response variable, but we may also be interested in correlations between the responses. To investigate these correlations we would have to fit the responses as a multivariate response vector.

Another area where multivariate modelling is useful is the treatment of missing data. Generally when we fit a univariate model with missing responses, the missing data have to be discarded unless we make some additional assumptions and impute values for them. In multivariate modelling we will often have incomplete response vectors but we can still use such data by imputing the missing responses using the correlations that have been found from the complete records (see later).

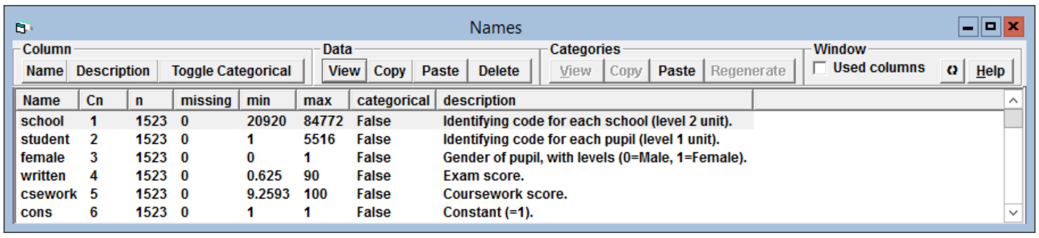
In this chapter we will firstly consider a dataset with two responses and complete records for every individual. This dataset is a subset of a larger dataset, which also includes individuals who have one or other response missing. We will then analyse the complete dataset. We finally look at a dataset with more variables and show how we can use a multivariate multilevel model to perform multiple imputation (Rubin, 1987). In this chapter we will consider continuous responses only but will consider how to deal with other response types in Chapter 19.

18.1 GCSE science data with complete records only

We will firstly consider a dataset of pupils' marks from the General Certificate of Secondary Education (GCSE) exams taken in 1989. The examination consisted of two parts, the first being a traditional written question paper (marked out of 160 but rescaled to create a score out of 100) and the second being coursework assignments (marked out of 108 but again rescaled to create a score out of 100). The dataset consists of data on 1905 students from 73 schools in England although we only have complete records on 1523 students. First we will open up the worksheet and look at the variable names:

- Select **Open Sample Worksheet** from the **File** menu.
- Select **gcsecomp1.ws** from the list and click on the **Open** button.

The variables will then appear as follows:



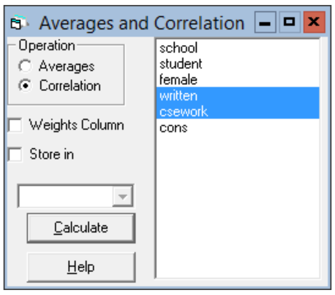
Name	Cn	n	missing	min	max	categorical	description
school	1	1523	0	20920	84772	False	Identifying code for each school (level 2 unit).
student	2	1523	0	1	5516	False	Identifying code for each pupil (level 1 unit).
female	3	1523	0	0	1	False	Gender of pupil, with levels (0=Male, 1=Female).
written	4	1523	0	0.625	90	False	Exam score.
csework	5	1523	0	9.2593	100	False	Coursework score.
cons	6	1523	0	1	1	False	Constant (=1).

Here we see the two response variables, **written** and **csework**, identifiers for the **student** and the **school** for each observation and one gender-based predictor variable **female**. As with analysis of univariate responses we can start by considering simple summary statistics and single level models.

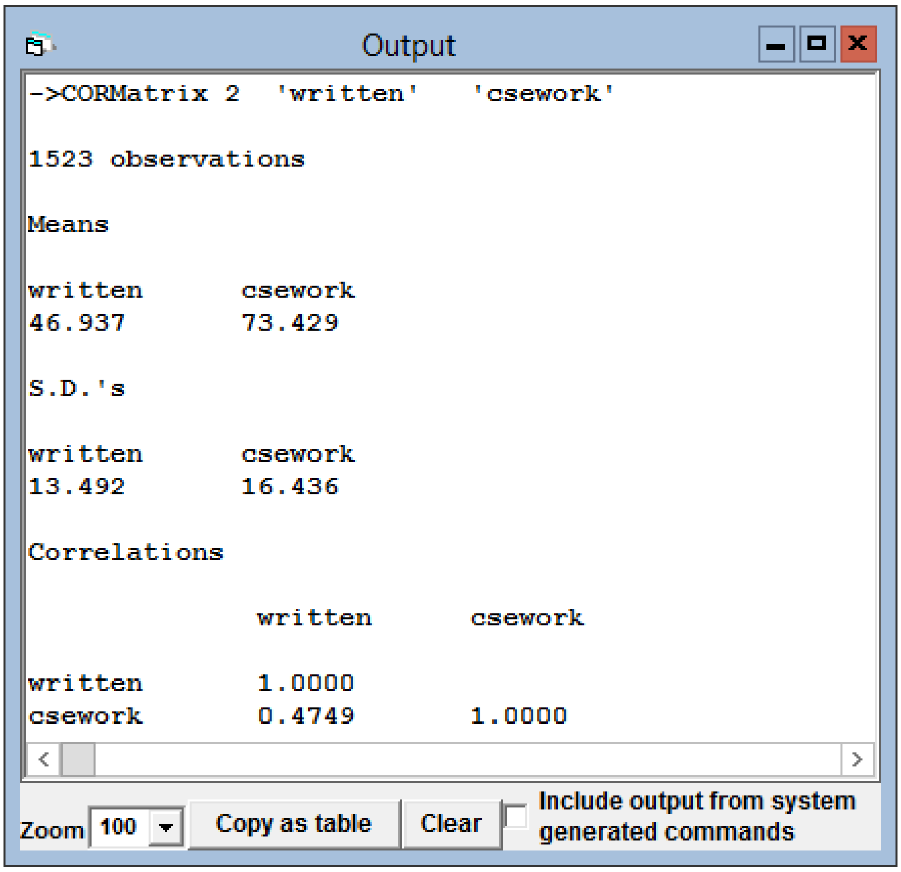
We will first look at some summary statistics for the two responses:

- Select **Average and Correlations** from the **Basic Statistics** menu.
- Select the **Correlation** button (as this gives averages as well).
- Select both **written** and **csework** (you can use the mouse and the Ctrl button).

The window should now look as follows:



If we now hit the **Calculate** button we will get the following estimates:



```
->CORMatrix 2  'written'   'csework'

1523 observations

Means

written       csework
46.937        73.429

S.D.'s

written       csework
13.492        16.436

Correlations

              written      csework

written       1.0000
csework       0.4749       1.0000
```

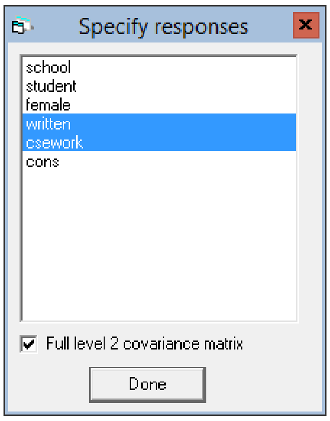
So here we see that the average coursework marks are higher than the written marks, coursework marks are more variable than written marks and there is a fairly high positive correlation (0.475) between the pairs of marks.

18.2 Fitting single level multivariate models

To fit multivariate models in MLwiN the software needs to create the correct format for the data. In earlier versions of MLwiN this was done via a special **Multivariate** window but now can be done directly via the **Equations** window. Perhaps the simplest multivariate model we could fit would simply replicate the summary statistics above and we will now show how to construct such a model.

- Select **Equations** from the **Model** menu.
- Click on the **Responses** button at the bottom of the window
- Click on **written** from the list that appears.
- Next click on **csework** from the list that appears.

The **Responses** window should now look as follows:



Full level 2 covariance matrix [✓]

- Click on the **Done** button on the **Responses** window.
- On the **Equations** window click on the response names to bring up the **Y variable** window.

The **Y variable** window will now appear and indicates that one level has been set up with identifiers in a column labelled **resp_indicator**. As explained in the User's Guide to MLwiN, the IGLS algorithm in MLwiN fits multivariate models by the clever trick of considering them as univariate models with no random variation at level 1 (an additional level which is created underneath the individual level). This can be achieved by transforming our original data by replication and pre-multiplication by indicator vectors that identify which variable is associated with each response. Note that this is done automatically by the software if we set up more than one response variable.

For example if we have the two equations

$$\begin{aligned}\mathbf{Written}_i &= \mathbf{Cons} \times \beta_1 + u_{1i} \\ \mathbf{Csework}_i &= \mathbf{Cons} \times \beta_2 + u_{2i}\end{aligned}$$

then we can stack the two response columns into one response column as follows:

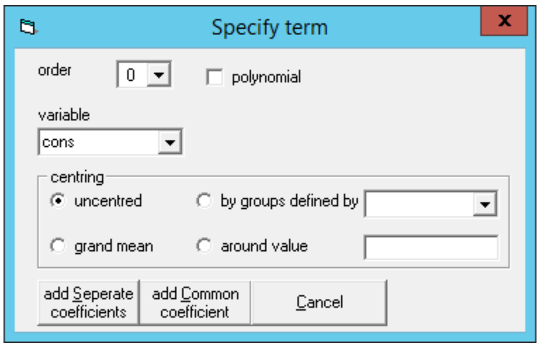
$$\mathbf{Resp}_{ir} = I(r=1)\times\mathbf{Cons}\times\beta_1+I(r=2)\times\mathbf{Cons}\times\beta_2+I(r=1)\times u_{1i}+I(r=2)\times u_{2i}$$

Here r is 1 for a written response and 2 for a coursework response and the function $I(x)$ is equal to 1 if x is true and 0 otherwise.

To set up a single-level multivariate model we now need to specify the individual level identifiers and the intercept terms in the **Equations** window (as we would for a univariate model):

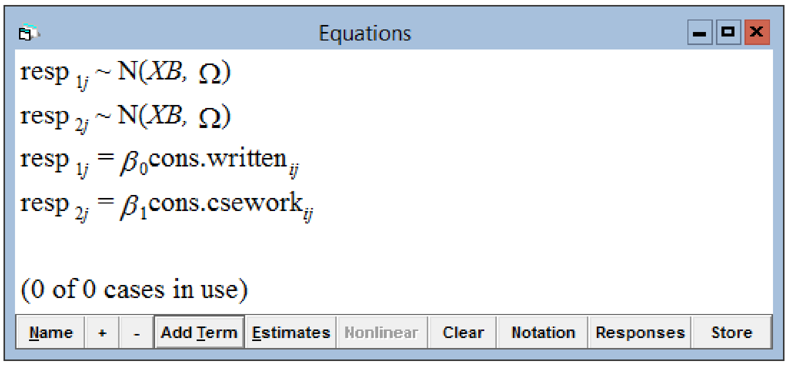
- On the **Y variable** window change the number of levels to **2-ij**.
- Select **student** for **level 2(j)**.
- Click on the **Done** button.
- On the **Equations** window, click on the **Add Term** button.
- Choose **cons** from the **variable** list on the **Add Term** window.

The **Specify term** window will then look as follows:



As you can see there are two options for adding terms into the multivariate model. Firstly, as we will use here, we can **add Separate coefficients** which will add one term to each response equation. Alternatively we can use the **add Common coefficient** option, which allows the user to specify which responses will share a common term. This is useful if we have several responses that we believe have the same relationship with a given predictor.

Click on the **add Separate coefficients** button and we will see the following:



$\text{resp}_{1j} \sim \text{N}(XB, \Omega)$

$\text{resp}_{2j} \sim \text{N}(XB, \Omega)$

$\text{resp}_{1j} = \beta_0\text{cons.written}_{ij}$

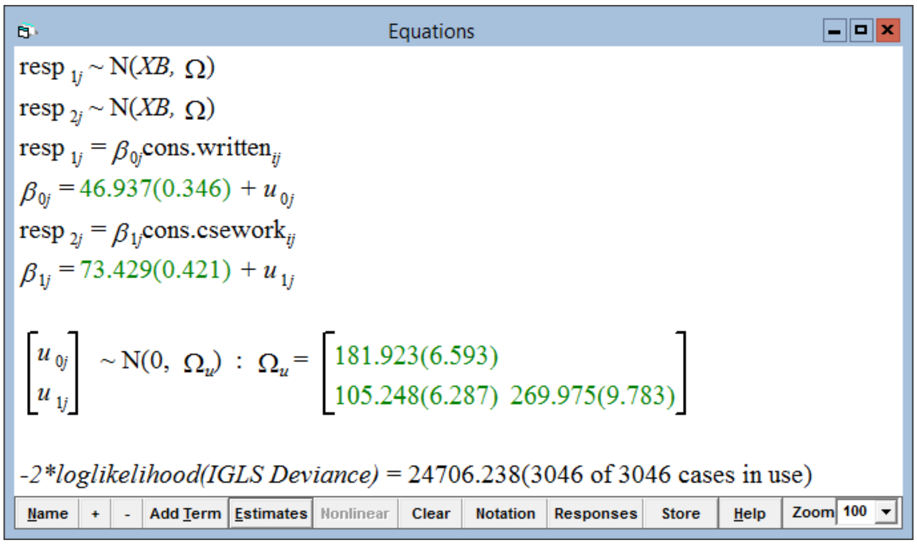
$\text{resp}_{2j} = \beta_1\text{cons.csework}_{ij}$

(0 of 0 cases in use)

We now need to specify the random part of the model:

- Click on the β_0 (**cons.written**) and select the **j(student_long)** tick box (keeping fixed effect selected) and then click on the **Done** button.
- Click on the β_1 (**cons.csework**) and select the **j(student_long)** tick box (keeping fixed effect selected) and then click on the **Done** button.
- Note that **student_long** is a column the software has created by repeating the student identifiers twice, once for each response.
- Click on the **Start** button.

This will produce (after clicking on the + and **Estimates** buttons twice) the following IGLS estimates:



$$\text{resp}_{1j} \sim \text{N}(XB, \Omega)$$
$$\text{resp}_{2j} \sim \text{N}(XB, \Omega)$$
$$\text{resp}_{1j} = \beta_{0j}\text{cons.written}_{ij}$$
$$\beta_{0j} = 46.937(0.346) + u_{0j}$$
$$\text{resp}_{2j} = \beta_{1j}\text{cons.csework}_{ij}$$
$$\beta_{1j} = 73.429(0.421) + u_{1j}$$
$$\begin{bmatrix} u_{0j} \\ u_{1j} \end{bmatrix} \sim \text{N}(0, \Omega_u) : \Omega_u = \begin{bmatrix} 181.923(6.593) & \\ 105.248(6.287) & 269.975(9.783) \end{bmatrix}$$

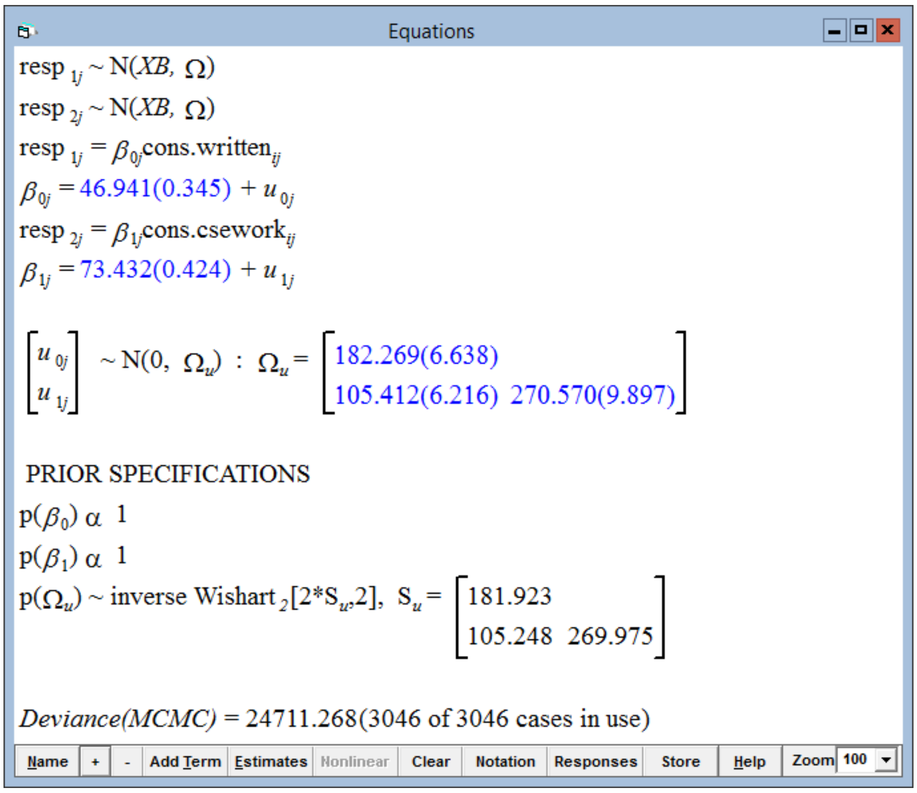
*-2*loglikelihood(IGLS Deviance)* = 24706.238(3046 of 3046 cases in use)

Here if we compare our results with the earlier summary statistics we see that the means are represented in the model by the fixed effects. The variance matrix in the model at level 2 will give variances that are approximately equal to the square of the standard deviations quoted earlier. Note however that IGLS is a maximum likelihood method and so the variances here are based on a variance formula using a divisor of n whilst RIGLS and the standard deviations earlier use a divisor of $n-1$ (i.e. we get exactly the same results as before if we switch to RIGLS). The covariance here can also be used along with the variance to give approximately the same estimate for the correlation quoted earlier.

If we now want to fit this model using MCMC we need to do the following:

- Change **Estimation method** to **MCMC**.
- Click on the **Start** button.

After the 5,000 iterations have run we get (after pressing the + button once again) the following estimates:



$\text{resp}_{1j} \sim \text{N}(XB, \Omega)$

$\text{resp}_{2j} \sim \text{N}(XB, \Omega)$

$\text{resp}_{1j} = \beta_{0j}\text{cons.written}_{ij}$

$\beta_{0j} = 46.941(0.345) + u_{0j}$

$\text{resp}_{2j} = \beta_{1j}\text{cons.csework}_{ij}$

$\beta_{1j} = 73.432(0.424) + u_{1j}$

$$\begin{bmatrix} u_{0j} \\ u_{1j} \end{bmatrix} \sim \text{N}(0, \Omega_u) : \Omega_u = \begin{bmatrix} 182.269(6.638) & \\ 105.412(6.216) & 270.570(9.897) \end{bmatrix}$$

PRIOR SPECIFICATIONS

$p(\beta_0) \propto 1$

$p(\beta_1) \propto 1$

$$p(\Omega_u) \sim \text{inverse Wishart}_2[2*S_u, 2], \; S_u = \begin{bmatrix} 181.923 & \\ 105.248 & 269.975 \end{bmatrix}$$

Deviance(MCMC) = 24711.268(3046 of 3046 cases in use)

Here we see that MCMC gives slightly larger variance estimates but this is mainly because the estimates are means rather than modes and the parameters have skew distributions. As with univariate models we can now use the DIC diagnostic for model comparison. The deviance formula for multivariate Normal models is:

$$\text{Deviance} = -\frac{N}{2}\log(2\pi) - \frac{1}{2}\log\left|\hat{\Sigma}_u\right| - \sum_{i=1}^{N}(y_i - \hat{y}_i)^T\hat{\Sigma}_u^{-1}(y_i - \hat{y}_i)$$

Selecting **MCMC/DIC diagnostic** from the **Model** menu we get the following:

Dbar	D(thetabar)	pD	DIC	
24711.27	24706.25	5.02	24716.29	
25099.52	25095.50	4.02	25103.54	(separate models)

Here as with other one level models the pD diagnostic corresponds almost exactly to the correct degrees of freedom. We can compare this model with a model that assumes no correlation between responses and hence separate models for each response and we see a large reduction of DIC.

18.3 Adding predictor variables

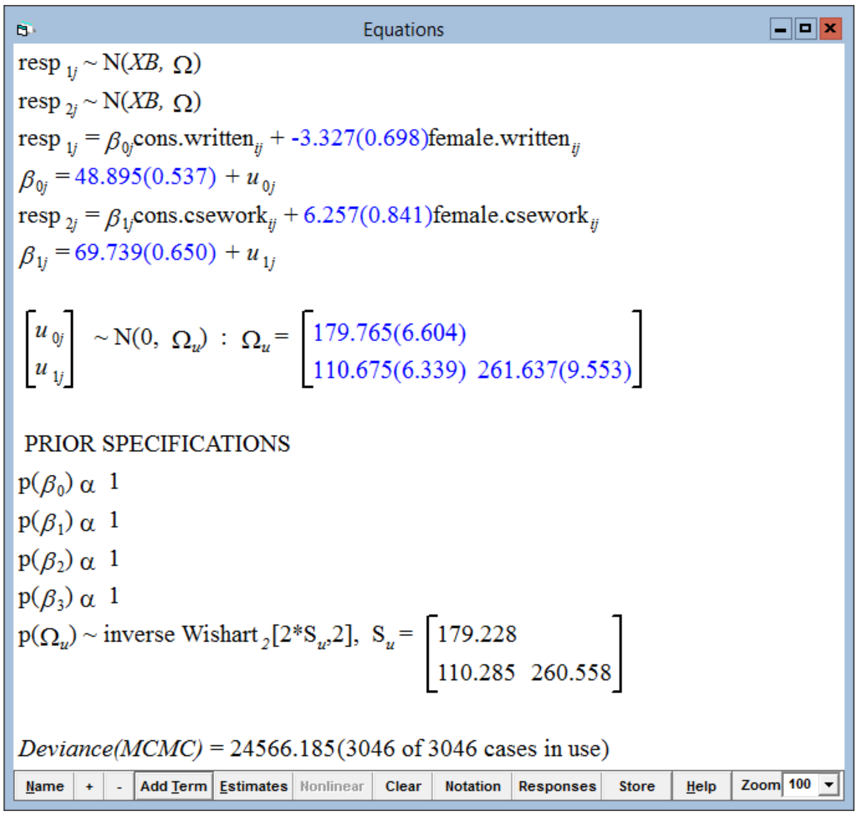
As with single response models we can now add predictor variables to our model as fixed effects. In this dataset we only have one additional predictor, the sex of the individual students. To add new variables into a multivariate model we need to use the **Add Term** window.

- Change **Estimation method** to **IGLS**.
- Click on the **Add Term** button on the **Equations** window.
- Select **female** from the variable list.
- Click on the **add Separate coefficients** button.

This will add the additional two fixed effects, one to each of the response equations. We can now run the model:

- Click on the **Start** button.
- Change **Estimation method** to **MCMC**.
- Click on the **Start** button.

After running for 5,000 iterations we get the following estimates:



$\text{resp}_{1j} \sim N(XB, \Omega)$

$\text{resp}_{2j} \sim N(XB, \Omega)$

$\text{resp}_{1j} = \beta_{0j}\text{cons.written}_{ij} + -3.327(0.698)\text{female.written}_{ij}$

$\beta_{0j} = 48.895(0.537) + u_{0j}$

$\text{resp}_{2j} = \beta_{1j}\text{cons.csework}_{ij} + 6.257(0.841)\text{female.csework}_{ij}$

$\beta_{1j} = 69.739(0.650) + u_{1j}$

$$\begin{bmatrix} u_{0j} \\ u_{1j} \end{bmatrix} \sim N(0,\ \Omega_u)\ :\ \Omega_u = \begin{bmatrix} 179.765(6.604) & \\ 110.675(6.339) & 261.637(9.553) \end{bmatrix}$$

PRIOR SPECIFICATIONS

$p(\beta_0)\ \alpha\ 1$

$p(\beta_1)\ \alpha\ 1$

$p(\beta_2)\ \alpha\ 1$

$p(\beta_3)\ \alpha\ 1$

$$p(\Omega_u) \sim \text{inverse Wishart}_2[2{*}S_u,2],\ S_u = \begin{bmatrix} 179.228 & \\ 110.285 & 260.558 \end{bmatrix}$$

Deviance(MCMC) = 24566.185(3046 of 3046 cases in use)

Here we see that girls do 6.3 points better on average at coursework but 3.3 points worse on average at the written paper. If we compare our model with the last model via the DIC diagnostic we see a significant reduction in DIC, which is to be expected given the two gender effects are both significantly larger than their standard errors.

Dbar	D(thetabar)	pD	DIC	
24566.19	24559.21	6.98	24573.16	(with gender effects)
24711.27	24706.25	5.02	24716.29	(without gender effects)

18.4 A multilevel multivariate model

We can now consider the effect of school attended on the exam score. As there are 73 schools we will fit the school effects as random terms and this results in a two level multivariate response model which is treated in MLwiN as a three level model (with the responses treated as an additional lower level). In the bivariate case the school attended can affect both responses and so we will have two school level effects for each school and these could be correlated. This will result in a 2×2 school level variance matrix. To fit this model we firstly need to return to the **Equations** window and define our additional level.

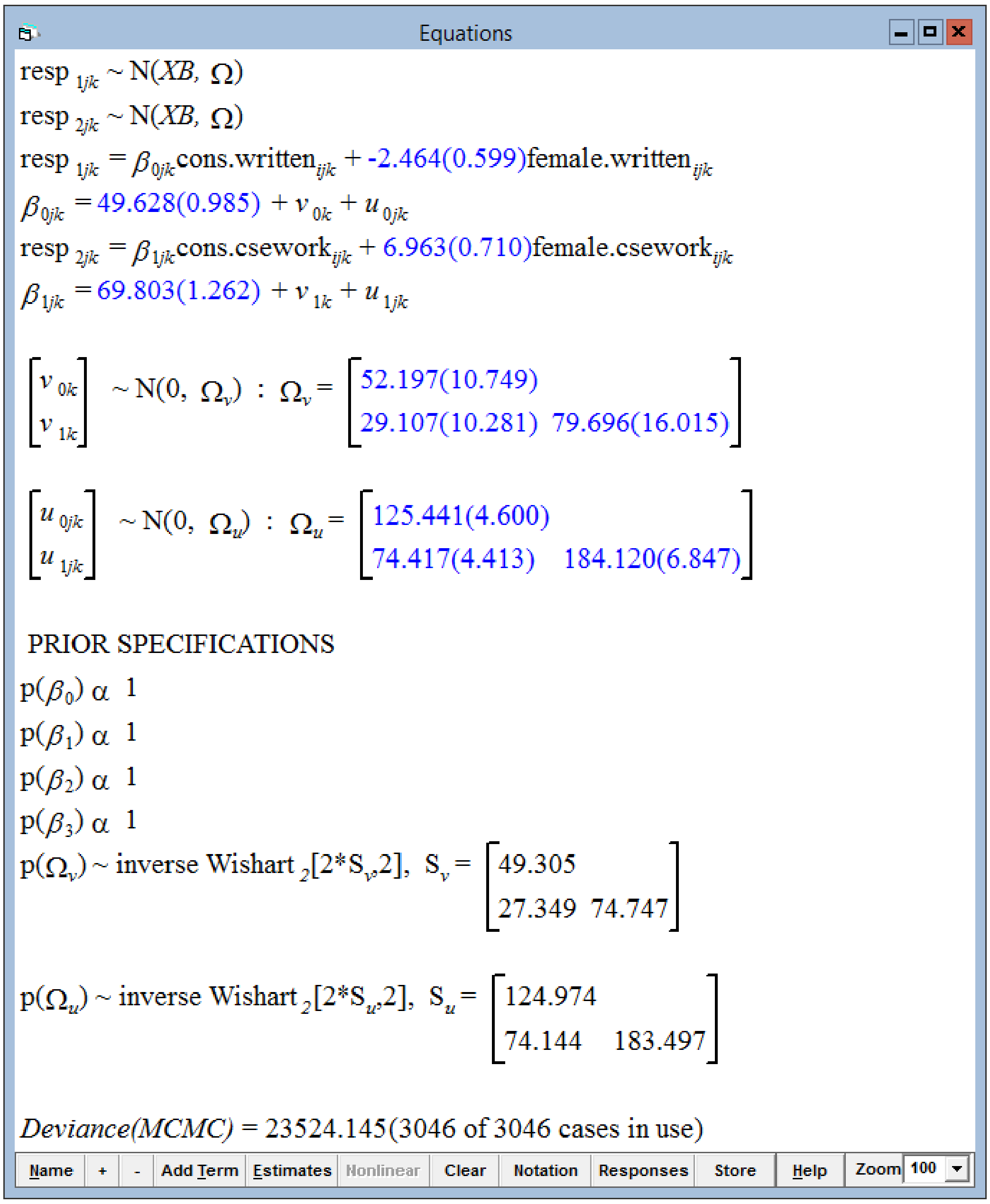
- Change **Estimation method** to **IGLS**
- Select the **Equations** window from the **Model** menu.
- Click on the response names to bring up the **Y variable** window.
- Select **3-ijk** as the number of levels.
- Select **school** from the **level 3(k)** pull down list.
- Click on the **Done** button.

We now need to add the two sets of random effects in the **Equations** window and run the model using firstly IGLS and then MCMC.

- Select **Equations** from the **Model** menu.
- Click on the β_{0j} (**cons_written**) and click on the **k(school_long)** tickbox.
- Click on the **Done** button.
- Click on the β_{1j} (**cons_csework**) and click on the **k(school_long)** tickbox.
- Click on the **Done** button.
- Click on the **Start** button.

- Change **Estimation method** to **MCMC**.
- Click on the **Start** button.

After running for 5,000 iterations we get the following estimates:



$\text{resp}_{1jk} \sim \text{N}(XB, \Omega)$

$\text{resp}_{2jk} \sim \text{N}(XB, \Omega)$

$\text{resp}_{1jk} = \beta_{0jk}\text{cons.written}_{ijk} + -2.464(0.599)\text{female.written}_{ijk}$

$\beta_{0jk} = 49.628(0.985) + v_{0k} + u_{0jk}$

$\text{resp}_{2jk} = \beta_{1jk}\text{cons.csework}_{ijk} + 6.963(0.710)\text{female.csework}_{ijk}$

$\beta_{1jk} = 69.803(1.262) + v_{1k} + u_{1jk}$

$$\begin{bmatrix} v_{0k} \\ v_{1k} \end{bmatrix} \sim \text{N}(0,\ \Omega_v)\ :\ \Omega_v = \begin{bmatrix} 52.197(10.749) & \\ 29.107(10.281) & 79.696(16.015) \end{bmatrix}$$

$$\begin{bmatrix} u_{0jk} \\ u_{1jk} \end{bmatrix} \sim \text{N}(0,\ \Omega_u)\ :\ \Omega_u = \begin{bmatrix} 125.441(4.600) & \\ 74.417(4.413) & 184.120(6.847) \end{bmatrix}$$

PRIOR SPECIFICATIONS

$p(\beta_0)\ \alpha\ 1$

$p(\beta_1)\ \alpha\ 1$

$p(\beta_2)\ \alpha\ 1$

$p(\beta_3)\ \alpha\ 1$

$$p(\Omega_v) \sim \text{inverse Wishart}_2[2{*}S_v,2],\ \ S_v = \begin{bmatrix} 49.305 & \\ 27.349 & 74.747 \end{bmatrix}$$

$$p(\Omega_u) \sim \text{inverse Wishart}_2[2{*}S_u,2],\ \ S_u = \begin{bmatrix} 124.974 & \\ 74.144 & 183.497 \end{bmatrix}$$

Deviance(MCMC) = 23524.145(3046 of 3046 cases in use)

Name + - Add Term Estimates Nonlinear Clear Notation Responses Store Help Zoom 100

If we were to compare the multilevel model to the single level model via the DIC diagnostic we will see the following:

Dbar	D(thetabar)	pD	DIC	
23524.14	23397.53	126.61	23650.76	(multilevel model)
24566.19	24559.21	6.98	24573.16	(single level model)

Here we see that the multilevel specification reduces the DIC by roughly 900 suggesting a much better model. The effective number of parameters (126.66) is slightly less than the 153 parameters in the model.

As with the univariate models in earlier chapters we have partitioned our unexplained variation into that which is due to the schools and that which is due to the students.

Here the school level explains 29.3% (52.197/(52.197+125.441)) of the remaining variation in the written scores and 30.2% (79.696/(79.696+184.120)) of the remaining variation in the coursework scores. There is also a fairly strong positive correlation between responses at the school level (0.45) suggesting that schools with high average coursework scores also have high average written scores.

We can investigate this further by looking at the school level residuals for this model.

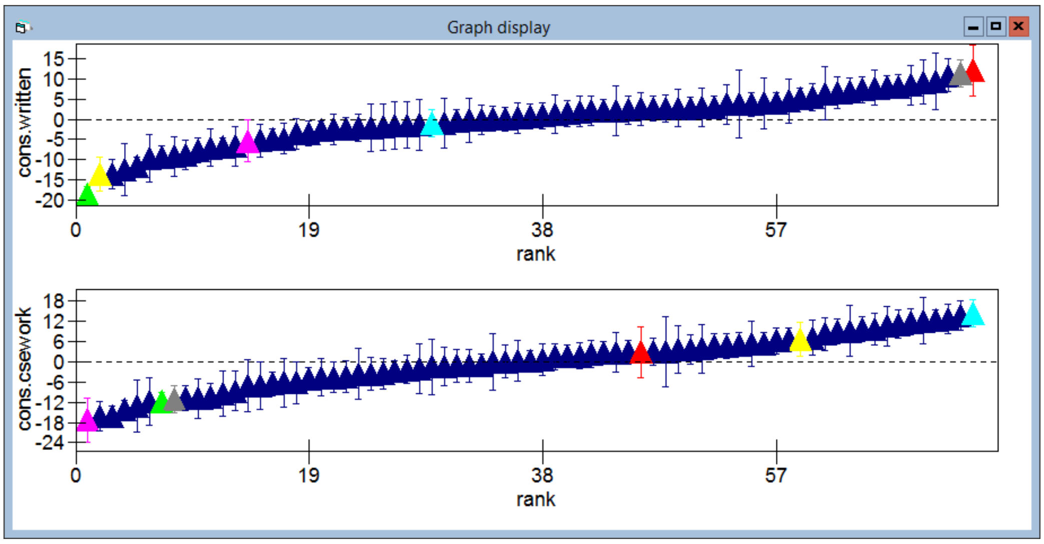
- Select **Residuals** from the **Model** menu.
- Select **3:school_long** from the **level** pull down list.
- Change the **SD multiplier** from 1.0 to 1.4.
- Click on the **Calc** button.

It should be noted that MLwiN may give some error messages here. If so simply click on the OK button. These messages refer to the deletion residuals, which are currently not calculated correctly when MCMC is used. If the tick for deletion residuals is removed on the ***Residuals*** *window the error messages will not appear.*

The school residuals have been stored in columns **c300** and **c301**. We can now look at 'caterpillar' plots of the residuals as we have seen in earlier chapters:

- Select the **Plot** tab on the **Residuals** window.
- Select **Residual +/- 1.4 sd x rank**.
- Click on the **Apply** button.

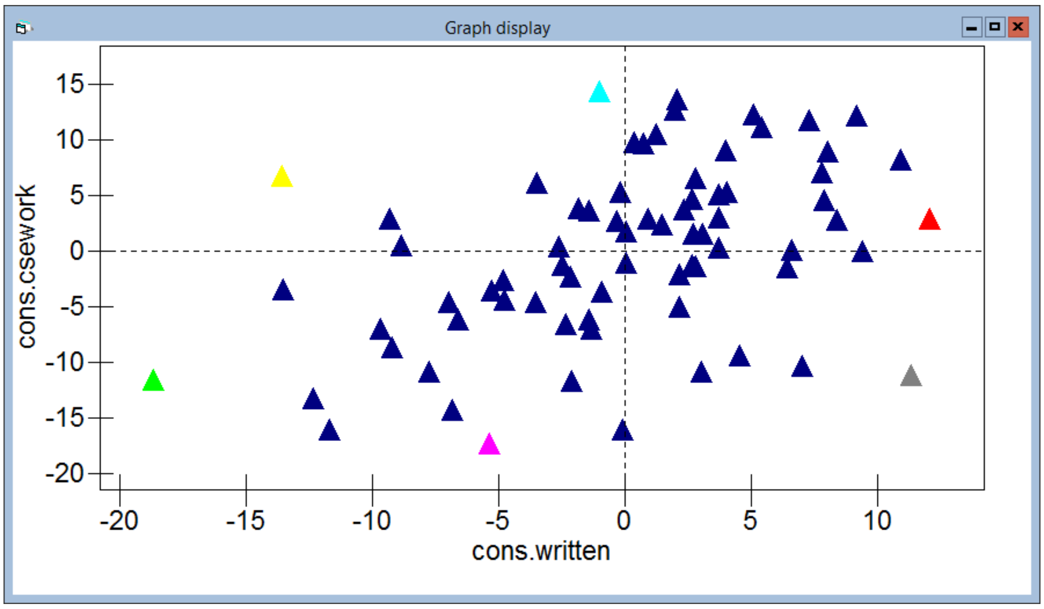
Two 'caterpillar' plots (minus the highlighting) will appear as shown below:



In the graphs we have highlighted (in various colours) the highest and lowest performing school for each response, and two other schools that are interesting as we will see in the next plot. As we have two residuals we can also look at pair-wise comparisons, which construct a 2 dimensional representation of the above caterpillar plots.

- Select the **Residuals** button in the **pair-wise** box.
- Click on the **Apply** button.

The following graph will then appear:



Here we see the school effects for both the written and coursework scores and we see why the two additional schools have been highlighted. School 22710 (highlighted yellow in the top left of the graph) has a below average written test effect but an above average coursework effect whilst school 67105 (highlighted grey in the bottom right of the graph) has an above average written score but a below average coursework. The reason these schools are interesting is that the written test score was externally marked whilst the coursework score was internally assessed but externally moderated. Thus it would be interesting to investigate these schools more closely to confirm whether the difference in scores is due to actual disparities between pupils' practical and examination abilities or due to differences between external and internal assessment.

18.5 GCSE science data with missing records

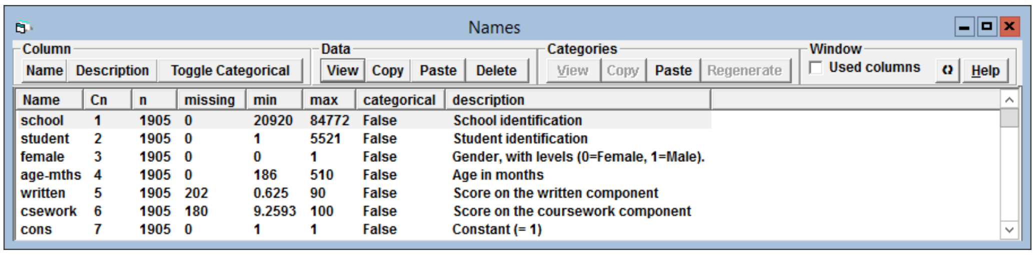
The analyses we have performed so far would be appropriate if we had complete records for all the data collected, but in our dataset we have other partial records that we have so far ignored. Ignoring missing data is dangerous because this can introduce biases into all the estimates in the model. We will now consider how to deal with missing data in a more sensible way. There are many techniques that deal with missing data and these can be split into two families: First imputation-based techniques that attempt to generate complete datasets before fitting models in the usual way to the imputed data. Secondly model-based techniques that include the missing data as a part of the model, and hence fit more complex models that account for the missing data.

Imputation-based techniques are usually simpler but may perform worse if they fail to account for the uncertainty in the missing data whilst model-based techniques may become impractical for more complex problems. In this example we will consider a model based MCMC algorithm for multivariate Normal models with missing responses. Then for our second example we will consider an imputation technique called 'multiple imputation' (Rubin, 1987) which gets around some of the lack of uncertainty in the missing data by generating several imputed datasets.

Firstly we will load up the complete data for the Science GCSE example:

- Select **Open Sample Worksheet** from the **File** menu.
- Select **gcsemv1.ws** from the list and click on the **Open** button.

The **Names** window will then appear as shown below. It should be noted here that the missing data has been coded globally (you can use the **Options/Numbers** window to do this with your own data) as missing rather than −1 as in the example in the User's Guide to MLwiN.



Name	Cn	n	missing	min	max	categorical	description
school	1	1905	0	20920	84772	False	School identification
student	2	1905	0	1	5521	False	Student identification
female	3	1905	0	0	1	False	Gender, with levels (0=Female, 1=Male).
age-mths	4	1905	0	186	510	False	Age in months
written	5	1905	202	0.625	90	False	Score on the written component
csework	6	1905	180	9.2593	100	False	Score on the coursework component
cons	7	1905	0	1	1	False	Constant (= 1)

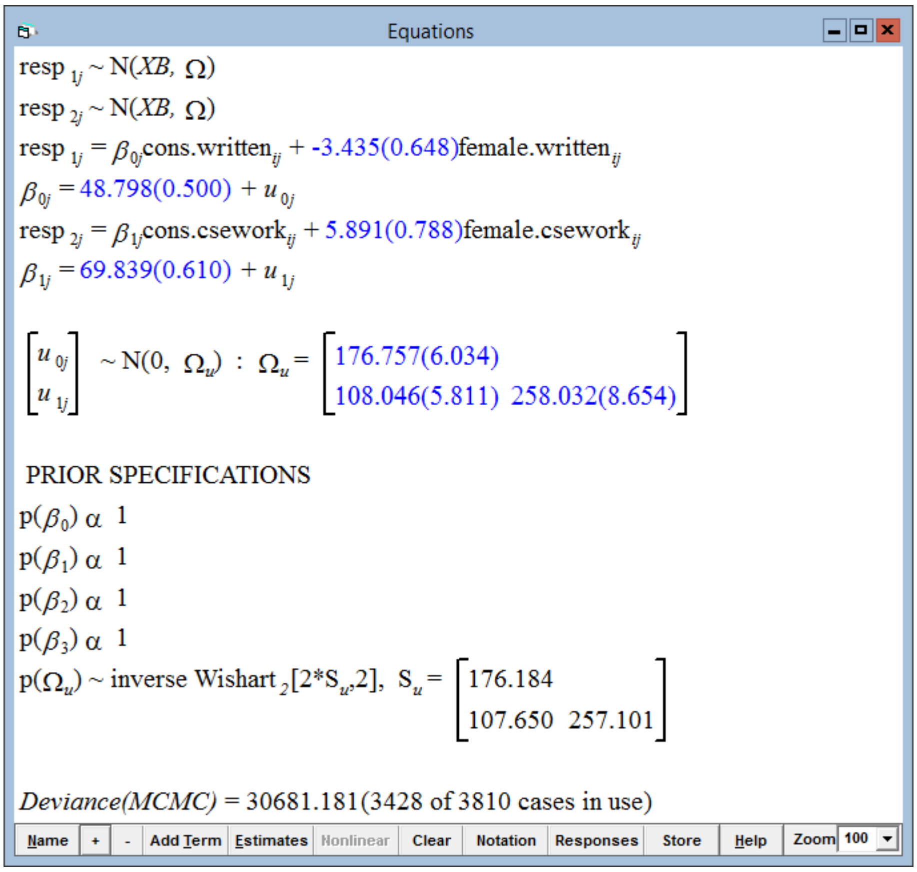
We can now repeat the earlier analyses using this complete dataset. We will ignore the simpler model and consider firstly the single level model with gender effects that we fitted earlier. We can set up this model in an identical way as described earlier so if you are unsure of how to do this refer back to the earlier sections. In MLwiN the IGLS and MCMC methods fit this model using different approaches which are both equivalent to assuming a 'missing at random' or MAR assumption (Rubin, 1976). The IGLS method, due to the clever trick of treating the multivariate problem as a special case of a univariate problem, can simply remove the missing rows in the longer data vector, which contains one row for each response. (See the User's Guide to MLwiN for more details).

The MCMC method considers the missing data as additional parameters in the model and assumes an independent uniform prior for each missing response. Then the missing records have Normal posterior distributions (multivariate Normal for individuals with more than one missing response) and the Gibbs sampling algorithm is extended to include an additional step that generates values for the missing data values at each iteration in the sampling.

To run the two approaches on our dataset we need to do the following:

- Set up the single level model with gender effects as described earlier.
- Click on the **Start** button.
- Change **Estimation method** to **MCMC**.
- Click on the **Start** button.

When the 5,000 iterations for the MCMC method have finished the Equations window will look as follows:



$\text{resp}_{1j} \sim N(XB, \Omega)$

$\text{resp}_{2j} \sim N(XB, \Omega)$

$\text{resp}_{1j} = \beta_{0j}\text{cons.written}_{ij} + -3.435(0.648)\text{female.written}_{ij}$

$\beta_{0j} = 48.798(0.500) + u_{0j}$

$\text{resp}_{2j} = \beta_{1j}\text{cons.csework}_{ij} + 5.891(0.788)\text{female.csework}_{ij}$

$\beta_{1j} = 69.839(0.610) + u_{1j}$

$$\begin{bmatrix} u_{0j} \\ u_{1j} \end{bmatrix} \sim N(0, \Omega_u) : \Omega_u = \begin{bmatrix} 176.757(6.034) & \\ 108.046(5.811) & 258.032(8.654) \end{bmatrix}$$

PRIOR SPECIFICATIONS

$p(\beta_0) \propto 1$

$p(\beta_1) \propto 1$

$p(\beta_2) \propto 1$

$p(\beta_3) \propto 1$

$$p(\Omega_u) \sim \text{inverse Wishart}_2[2*S_u, 2], \quad S_u = \begin{bmatrix} 176.184 & \\ 107.650 & 257.101 \end{bmatrix}$$

Deviance(MCMC) = 30681.181(3428 of 3810 cases in use)

Notice that we now have 3428 responses observed and 382 responses missing out of our total of 3810 responses. The results we see here are fairly similar to those for the complete records only, although the female difference effects have changed from 6.25 to 5.89 on coursework and −3.32 to −3.43 on written examination. If we look at the DIC diagnostic for this model we get the following:

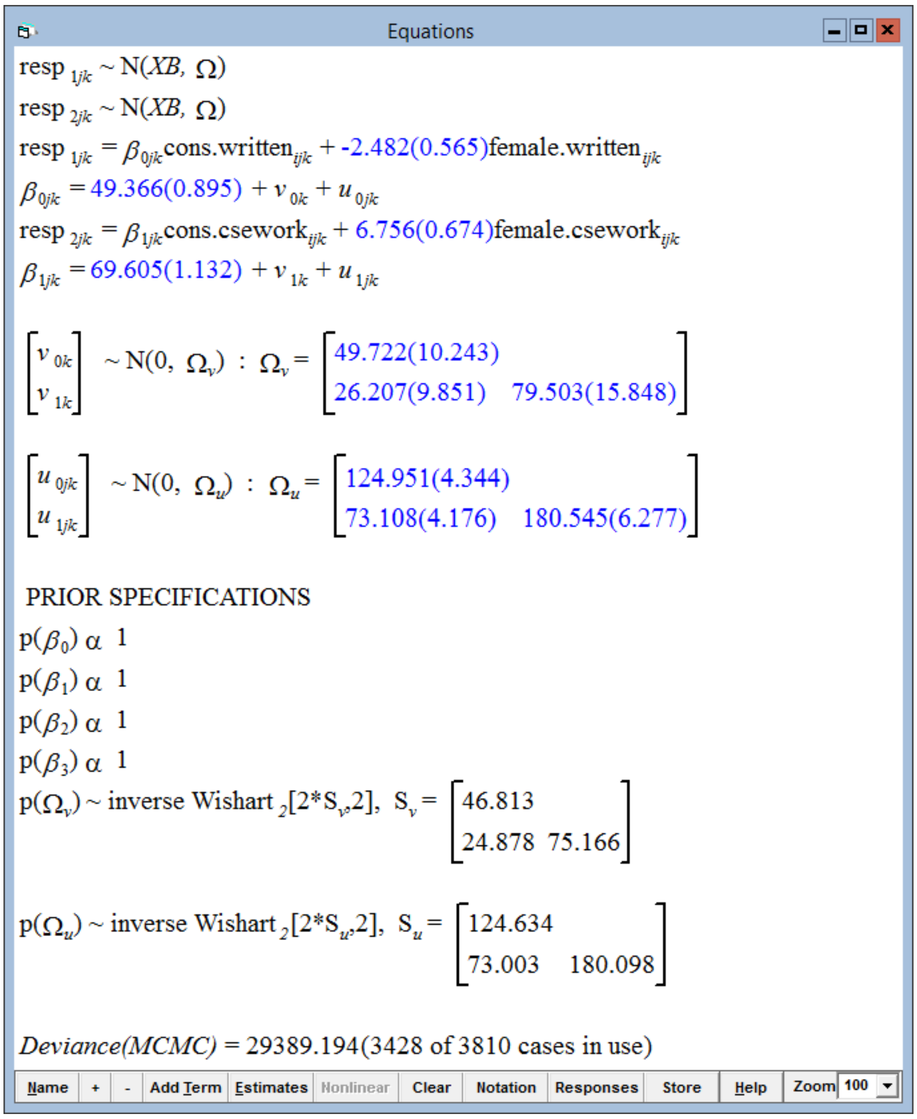
Dbar	D(thetabar)	pD	DIC
30681.18	30292.23	388.95	31070.13

We cannot compare this diagnostic with other previous models as we now effectively have a new dataset. However what is interesting is that the effective number of parameters (pD) is approximately 389, which equals the number of missing responses (382) plus the number of parameters (7). So we can see clearly that the MCMC algorithm treats missing data as additional parameters in the model. Note that an alternative here would be to reformulate the deviance function to marginalize out the missing data and this would give different pD and DIC estimates.

We can now look again at the multilevel model considered earlier with the complete cases:

- Change **Estimation method** to **IGLS**.
- Select **Equations** from the **Model** menu and click on the response names to bring up the **Y variable** window.
- Select **3-ijk** for the number of levels.
- Select **school** from the **level 3(k)** pull down list.
- Click on the **Done** button.
- Click on the β_{0j} (**cons_written**) and click on the **k(school_long)** tickbox.
- Click on the β_{1j} (**cons_csework**) and click on the **k(school_long)** tickbox.
- Click on the **Start** button.
- Change **Estimation method** to **MCMC**.
- Click on the **Start** button.

After running this model we get the following results, which are again similar to the estimates for the complete case data:



$$\text{resp}_{1jk} \sim \text{N}(XB,\ \Omega)$$
$$\text{resp}_{2jk} \sim \text{N}(XB,\ \Omega)$$
$$\text{resp}_{1jk} = \beta_{0jk}\text{cons.written}_{ijk} + -2.482(0.565)\text{female.written}_{ijk}$$
$$\beta_{0jk} = 49.366(0.895) + v_{0k} + u_{0jk}$$
$$\text{resp}_{2jk} = \beta_{1jk}\text{cons.csework}_{ijk} + 6.756(0.674)\text{female.csework}_{ijk}$$
$$\beta_{1jk} = 69.605(1.132) + v_{1k} + u_{1jk}$$

$$\begin{bmatrix} v_{0k} \\ v_{1k} \end{bmatrix} \sim \text{N}(0,\ \Omega_v)\ :\ \Omega_v = \begin{bmatrix} 49.722(10.243) & \\ 26.207(9.851) & 79.503(15.848) \end{bmatrix}$$

$$\begin{bmatrix} u_{0jk} \\ u_{1jk} \end{bmatrix} \sim \text{N}(0,\ \Omega_u)\ :\ \Omega_u = \begin{bmatrix} 124.951(4.344) & \\ 73.108(4.176) & 180.545(6.277) \end{bmatrix}$$

PRIOR SPECIFICATIONS

$$p(\beta_0)\ \alpha\ 1$$
$$p(\beta_1)\ \alpha\ 1$$
$$p(\beta_2)\ \alpha\ 1$$
$$p(\beta_3)\ \alpha\ 1$$
$$p(\Omega_v) \sim \text{inverse Wishart}_2[2*S_v,2],\ S_v = \begin{bmatrix} 46.813 & \\ 24.878 & 75.166 \end{bmatrix}$$

$$p(\Omega_u) \sim \text{inverse Wishart}_2[2*S_u,2],\ S_u = \begin{bmatrix} 124.634 & \\ 73.003 & 180.098 \end{bmatrix}$$

Deviance(MCMC) = 29389.194(3428 of 3810 cases in use)

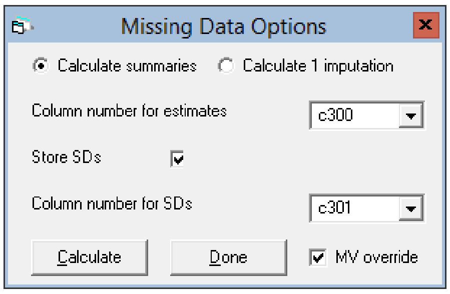
The DIC diagnostic again shows a great improvement in fitting a multilevel model:

Dbar	D(thetabar)	pD	DIC	
29389.19	28878.14	511.05	29900.25	(multilevel model)
30681.18	30292.23	388.95	31070.13	(single level model)

The MCMC method is generating values at each iteration for the missing data and we may be interested in the values generated. Currently MLwiN does not allow the user to store the chains of these missing values but summary statistics are available.

- Select **MCMC/Missing data** from the **Model** menu.
- Select the tickbox to **store SDs** (in **c301**).

The window should then look as follows:



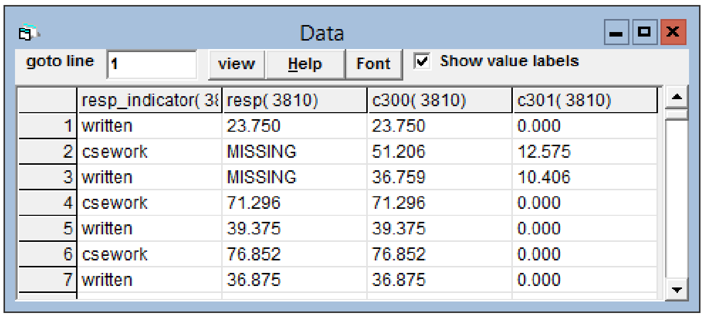
[✓ Calculate summaries] Calculate 1 imputation

Column number for estimates	c300
Store SDs	✓
Column number for SDs	c301
MV override	✓

Now click on the **Calculate** button and then the **Done** button to store the results and close the window.

- Select **View or Edit Data** from the **Data Manipulation** menu.
- Click on the **View** button.
- Select columns **resp_indicator**, **resp**, **c300**, and **c301**, which are lower down the list.
- Click on the **OK** button and resize the window to see all four columns.

The window should look as follows:



	resp_indicator(3§	resp(3810)	c300(3810)	c301(3810)
1	written	23.750	23.750	0.000
2	csework	MISSING	51.206	12.575
3	written	MISSING	36.759	10.406
4	csework	71.296	71.296	0.000
5	written	39.375	39.375	0.000
6	csework	76.852	76.852	0.000
7	written	36.875	36.875	0.000

Here we see that the first individual had a missing coursework response, which on average in the chain has estimate 51.2 with a standard deviation of 12.6. The second person has a missing written exam score which has predicted value 36.8 with a standard deviation of 10.4. Currently MLwiN only allows an MAR assumption for the response values although it is hoped that in future releases we will include options that allow informative prior distributions for missing data. Note that there are other modelling techniques for handling non MAR data but we shall not discuss these here.

18.6 Imputation methods for missing data

As mentioned earlier there are two families of techniques that can be used for missing data; imputation and model based methods. The model-based methods as we saw in the last example make special provisions in the model for the missing data. The imputation-based methods however are techniques that are used prior to modelling. We can think in fact of imputation techniques as being two stage techniques where the first stage involves imputing a complete dataset and the second stage involves modelling using this complete dataset.

The imputation step can be deterministic, for example simple imputation techniques involve substituting missing data with a determined value. A simple example of this determined value is the mean for the variable. Alternatively a model can be created for the variables to be imputed, for example a regression model and this model can then be used to generate predicted values to replace the missing values. Of course the model will give a point estimate and a standard error for each missing observation and so we could use an alternative approach of generating a random draw from the predictive distribution (under the imputation model) of the missing observation. This will then produce a stochastic imputation procedure.

The disadvantage of imputation-based methods is that once the complete dataset is generated all uncertainty about the missing values is lost. Multiple imputation (Rubin, 1987) aims to bridge the gap between standard imputation-based methods and model-based methods by using a stochastic imputation procedure several times. Then we can perform our modelling on each of these N imputed complete datasets and use for each parameter estimate an average of the N estimates produced by the datasets, along with a standard error that is constructed from a combination of the variance for each dataset and the variance between the datasets. So for a parameter θ we have an estimate

$$\hat{\theta} = \frac{1}{N}\sum_{i=1}^{N}\hat{\theta}_i, \quad \text{with variance } V(\hat{\theta}) = \bar{V} + (1 + N^{-1})B$$

$$\text{where } \bar{V} = \frac{1}{N}\sum_{i=1}^{\text{N}} V(\hat{\theta}_i) \text{ and } B = \frac{1}{N-1}\left(\hat{\theta}_i - \hat{\theta}\right)^2$$

This approach would be equivalent to a model based approach if N was very large but even an N as small as 5 will give a reasonable approximation.

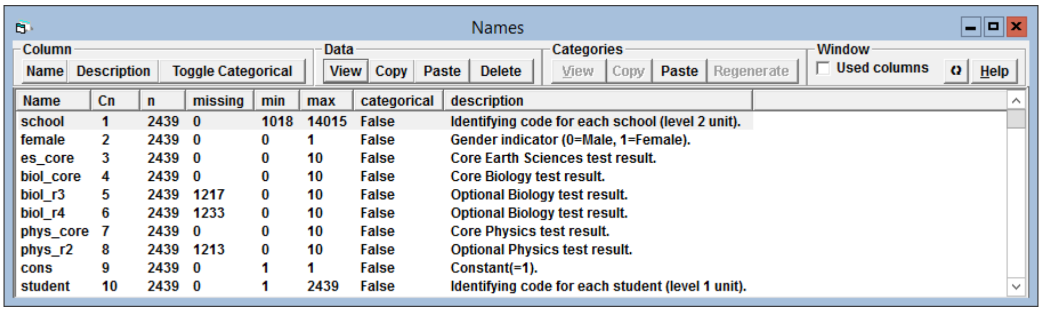
For more details on missing data and multiple imputation we recommend Schafer (1996). In this section we will simply show how to generate imputed datasets using multivariate multilevel models in MLwiN. The effectiveness of the imputation procedure will depend on the imputation model used. Typically single level multivariate normal models that take account of correlation between variables are used in multiple imputation. However it may be more appropriate to use a multilevel multivariate model as an imputation model (note that such models have been used independently in Schafer, 1997).

18.7 Hungarian science exam dataset

This dataset was analysed in Goldstein (2003) and is part of the Second International Science Survey carried out in 1984. The data we are using is the component from Hungary and consists of marks from six tests taken by 2,439 students in 99 schools. Of the six papers, three are core papers in earth sciences, biology and physics which were taken by all students whilst the other three are optional papers, two in biology and one in physics of which no student took all three. Although these tests were marked out of different totals ranging from 4 to 10, Goldstein (2003) converted each test to have scores between 0 and 1. Here we will use instead scores between 0 and 10 as these are closer to the original marks.

- Select **Open Sample Worksheet** from the **File** menu.
- Select **hungary1.ws** from the list and click on the **Open** button.

The variables in the dataset are as follows:



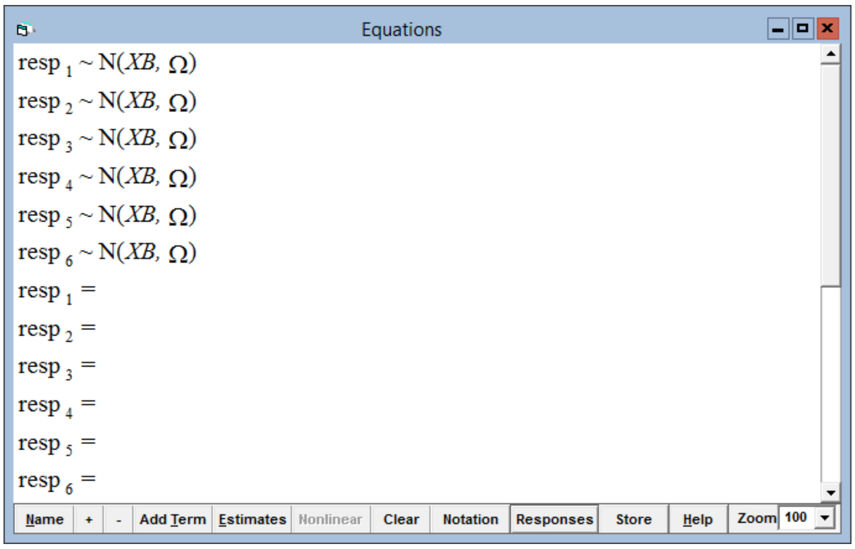
Name	Cn	n	missing	min	max	categorical	description
school	1	2439	0	1018	14015	False	Identifying code for each school (level 2 unit).
female	2	2439	0	0	1	False	Gender indicator (0=Male, 1=Female).
es_core	3	2439	0	0	10	False	Core Earth Sciences test result.
biol_core	4	2439	0	0	10	False	Core Biology test result.
biol_r3	5	2439	1217	0	10	False	Optional Biology test result.
biol_r4	6	2439	1233	0	10	False	Optional Biology test result.
phys_core	7	2439	0	0	10	False	Core Physics test result.
phys_r2	8	2439	1213	0	10	False	Optional Physics test result.
cons	9	2439	0	1	1	False	Constant(=1).
student	10	2439	0	1	2439	False	Identifying code for each student (level 1 unit).

Here we see the two level identifiers, **school** and **student**, the six test scores that are stored in columns **c3–c8** and one additional predictor **female** which indicates the sex of each student. As with the earlier example we could try and fit several different models to the dataset and compare the DIC diagnostic to find the 'best' model, here however we skip this stage and fit the model that Goldstein fits in his book.

To set up the model we first need to set up the model structure using the multivariate window.

- Select **Equations** from the **Model** menu.
- Click on the **Responses** button.
- On the list that appears click on **es_core**, **biol_core**, **biol_r3**, **biol_r4**, **phys_core** and **phys_r2**.
- Click on the **Done** button.

This will set up the responses and the **Equations** window should look as follows:



$\text{resp}_1 \sim \text{N}(XB, \Omega)$
$\text{resp}_2 \sim \text{N}(XB, \Omega)$
$\text{resp}_3 \sim \text{N}(XB, \Omega)$
$\text{resp}_4 \sim \text{N}(XB, \Omega)$
$\text{resp}_5 \sim \text{N}(XB, \Omega)$
$\text{resp}_6 \sim \text{N}(XB, \Omega)$
$\text{resp}_1 =$
$\text{resp}_2 =$
$\text{resp}_3 =$
$\text{resp}_4 =$
$\text{resp}_5 =$
$\text{resp}_6 =$

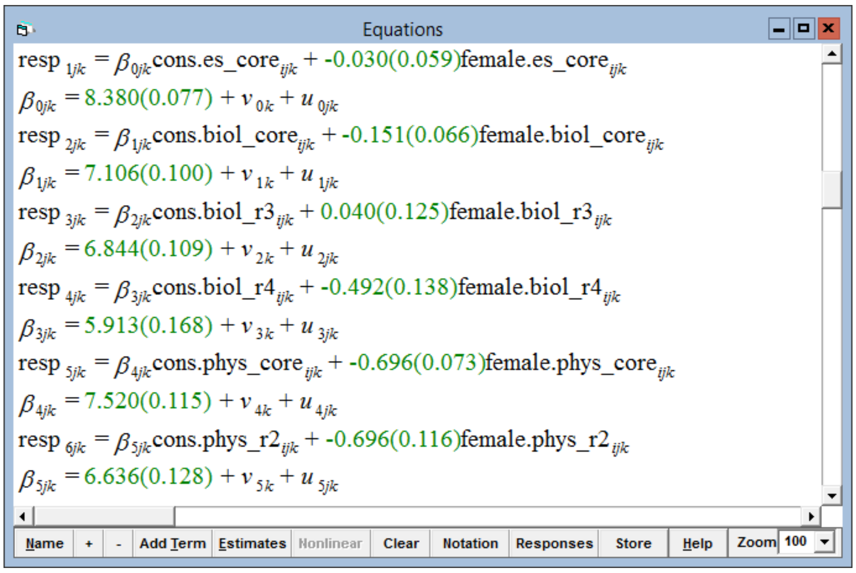
We now need to specify the level identifier variables and the predictor variables:

- Click on the responses in the **Equations** window.
- On the **Y variable** window that appears select **3-ijk** as the number of levels.
- Select **student** from the **level 2(j)** pull down list.
- Select **school** from the **level 3(k)** pull down list.
- Click on the **Done** button.
- Click on the **Add Term** button.
- Select **cons** from the **variable** list and click on the **add Separate coefficients** button.
- Click on the **Add Term** button.
- Select **female** from the **variable** list and click on the **add Separate coefficients** button.

We have now added the fixed effects into the model and we now need to add the random effects.

- Click on β_0 (**cons_es_core**) and select the **j(student_long)** and **k(school_long)** tick boxes (keeping fixed effect selected).
- Click on the **Done** button.
- Repeat this for the other constants, β_1, β_2, β_3, β_4 and β_5.
- Click on the **Start** button to run the model using IGLS.

After the IGLS model converges you should get fixed estimates as shown below. Note that due to the rescaling of the dataset these estimates are ten times the estimates in Goldstein (2003).



$\text{resp}_{1ijk} = \beta_{0ijk}\text{cons.es\_core}_{ijk} + \text{-}0.030(0.059)\text{female.es\_core}_{ijk}$

$\beta_{0jk} = 8.380(0.077) + v_{0k} + u_{0jk}$

$\text{resp}_{2ijk} = \beta_{1ijk}\text{cons.biol\_core}_{ijk} + \text{-}0.151(0.066)\text{female.biol\_core}_{ijk}$

$\beta_{1jk} = 7.106(0.100) + v_{1k} + u_{1jk}$

$\text{resp}_{3ijk} = \beta_{2ijk}\text{cons.biol\_r3}_{ijk} + 0.040(0.125)\text{female.biol\_r3}_{ijk}$

$\beta_{2jk} = 6.844(0.109) + v_{2k} + u_{2jk}$

$\text{resp}_{4ijk} = \beta_{3ijk}\text{cons.biol\_r4}_{ijk} + \text{-}0.492(0.138)\text{female.biol\_r4}_{ijk}$

$\beta_{3jk} = 5.913(0.168) + v_{3k} + u_{3jk}$

$\text{resp}_{5ijk} = \beta_{4ijk}\text{cons.phys\_core}_{ijk} + \text{-}0.696(0.073)\text{female.phys\_core}_{ijk}$

$\beta_{4jk} = 7.520(0.115) + v_{4k} + u_{4jk}$

$\text{resp}_{6ijk} = \beta_{5ijk}\text{cons.phys\_r2}_{ijk} + \text{-}0.696(0.116)\text{female.phys\_r2}_{ijk}$

$\beta_{5jk} = 6.636(0.128) + v_{5k} + u_{5jk}$

We will now consider fitting this model using MCMC. Although this model is itself interesting we are using it purely as an imputation model and are more interested in the missing values imputed than the fixed effect and variance estimates. To use multiple imputation it is desirable to have independent samples from the posterior distributions of the missing data. Consequently we will run for a burn-in of 500 as usual and then store the values every thousand iterations. To get our first sample we will do the following:

- Change **Estimation method** to **MCMC** and open the **Estimation Control** window.
- Change the **Monitoring Chain Length** to 1,000.
- Click on the **Start** button.

After the estimation finishes (which may take a couple of minutes) we need to then do the following:

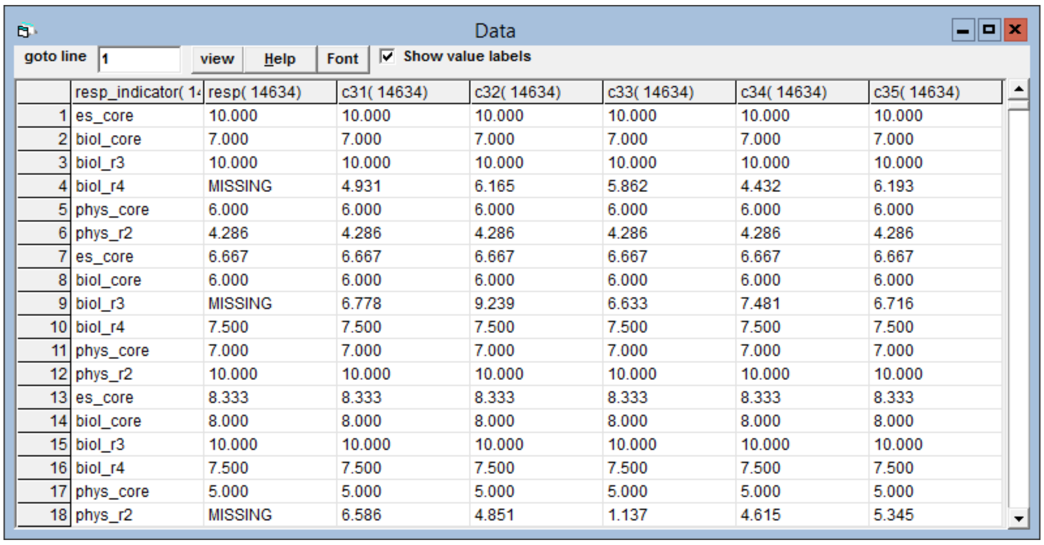
- Select **MCMC/Missing Data** from the **Model** menu.
- Select the **Calculate 1 imputation** button and change the output column to **c31**.
- Click on the **Calculate** button.

This will generate the first imputed dataset. It should be noted here that the values stored are the actual chain values after 1000 iterations and NOT the means of the sets of 1000 values. We can now repeat the procedure four more times by in turn changing the **Monitoring Chain Length** to 2,000, 3,000, 4,000 and 5,000 iterations and clicking on the **More** button. Then each time we can change the **Output column** on the **Missing Data** screen to **c32**, **c33**, **c34** and **c35** respectively and generate an imputed dataset.

It should at this point be noted that we could have written a macro to generate these datasets that would use the MCMC and DAMI commands (see the Command manual). After generating the 5 datasets we can view the results in the data window:

- Select **View or edit data** from the **Data Manipulation** menu.
- Click on the **view** button.
- Select columns **resp_indicator**, **resp**, **c31**, **c32**, **c33**, **c34**, and **c35**.
- Click on the **OK** button and resize the window.

The data will then look as follows:



goto line 1 ✓ Show value labels

	resp_indicator(1	resp(14634)	c31(14634)	c32(14634)	c33(14634)	c34(14634)	c35(14634)
1	es_core	10.000	10.000	10.000	10.000	10.000	10.000
2	biol_core	7.000	7.000	7.000	7.000	7.000	7.000
3	biol_r3	10.000	10.000	10.000	10.000	10.000	10.000
4	biol_r4	MISSING	4.931	6.165	5.862	4.432	6.193
5	phys_core	6.000	6.000	6.000	6.000	6.000	6.000
6	phys_r2	4.286	4.286	4.286	4.286	4.286	4.286
7	es_core	6.667	6.667	6.667	6.667	6.667	6.667
8	biol_core	6.000	6.000	6.000	6.000	6.000	6.000
9	biol_r3	MISSING	6.778	9.239	6.633	7.481	6.716
10	biol_r4	7.500	7.500	7.500	7.500	7.500	7.500
11	phys_core	7.000	7.000	7.000	7.000	7.000	7.000
12	phys_r2	10.000	10.000	10.000	10.000	10.000	10.000
13	es_core	8.333	8.333	8.333	8.333	8.333	8.333
14	biol_core	8.000	8.000	8.000	8.000	8.000	8.000
15	biol_r3	10.000	10.000	10.000	10.000	10.000	10.000
16	biol_r4	7.500	7.500	7.500	7.500	7.500	7.500
17	phys_core	5.000	5.000	5.000	5.000	5.000	5.000
18	phys_r2	MISSING	6.586	4.851	1.137	4.615	5.345

Here we see the data for the first three individuals in the dataset. As can be seen they each have one missing paper and the values for these papers have been generated from the respective posterior distribution. It should be noted at this point that in this model we are treating marks as continuous Normally distributed variables. This of course means that the imputed marks will follow these Normal distributions rather than the smaller set of values possible in the test. We are actively researching methods to impute ordered categorical data, although Schafer (1996) considers this subject in a non-multilevel context in great detail. Recent work under the REALCOM project has extended the missing data capabilities of MLwiN but this is described in its own documentation.

We will return to this dataset in Chapter 20 when we consider multilevel factor analysis modelling. In the next chapter we continue looking at multivariate responses and consider mixed response models and correlated residuals/patterned variance matrices.

Chapter learning outcomes

⋆ How to fit multivariate response models using MCMC in MLwiN.

⋆ How MLwiN deals with missing data.

⋆ How to fit multilevel multivariate models in MLwiN.

⋆ How to compare multivariate models using the DIC diagnostic.

⋆ How to generate datasets for multiple imputation in MLwiN.

Chapter 19

Mixed Response Models and Correlated Residuals

In the last chapter we introduced modelling of multiple responses and concentrated on responses that are all continuous variables. We assumed that these responses have a multivariate Normal distribution with unknown mean vector and variance matrix. In this chapter we will consider two extensions to such a model. Firstly we will consider how to fit mixtures of Normal and Bernoulli distributed variables. We will consider an example from education where we have a continuous English exam mark and a dichotomous behavioural measure. We will introduce some structure into the lowest level variance matrix to cater for the Bernoulli responses. Secondly we will consider more generally introducing specific structures into the variance matrix at the lowest level in particular for use with repeated measures. Here we consider an animal growth dataset and consider how to include auto-correlated residuals at the lowest level. The topic of MCMC modelling of constrained variance matrices is covered in Browne (2006).

19.1 Mixed response models

In the last chapter we concentrated on responses that are continuous variables. However as we saw in Chapters 10–13 not all response variables have continuous distributions as some responses may be counts or dichotomous variables. When we have mixtures of response types (as with models with all continuous responses) we wish to model correlation between the responses in addition to adjusting for predictors. This is difficult as we have different distributional assumptions for our different responses. The iterative procedures used in MLwiN have to use quasi-likelihood methods when some of the responses are non-Normal and this involves linearizing the non-Normal responses. This will result in estimates for covariances (correlations) at the lowest level, although what form the joint distribution of the responses at this level has is not known.

If we use MCMC estimation we can currently only fit mixtures of Normal and Binomial responses, and then only with a probit link for the Binomial. This is because only this combination of responses results in an identifiable distribution for the residuals at the lowest level. Other response combinations can be handled by making the (often unrealistic) assumption of independence at the bottom level, although not currently using the MCMC options in MLwiN.

The reason that the probit link Bernoulli and Normal combination can be fitted stems again from the thresholding idea used in Albert & Chib (1993) as discussed in Chapter 10. This idea for the multivariate case is also discussed for single level models in Chib & Hamilton (2000). Basically we can treat any Bernoulli responses as thresholded Normal responses with threshold 0 and variance 1. Thus if the observed response is 0, then the underlying Normal response is negative and if the observed response is 1, the underlying Normal response is positive.

Given we can sample these unknown continuous responses as an additional step in a Gibbs sampling algorithm then combining these responses with the other observed continuous responses results in all responses being continuous. This means we can then make a multivariate Normal assumption for the responses with the constraints that the variance for any of the unknown responses equals 1. This means that although, as in Chapter 10 we can now use Gibbs sampling to update both the fixed effects and the residuals we now have to use Metropolis sampling to update the elements of the lowest level variance matrix.

For generality we use the technique used in Browne (2006) of univariate Normal Metropolis updates for the unconstrained elements of the variance matrix. This technique involves using random walk Normal proposals for each parameter in turn and after producing a proposed new parameter value checking that it satisfies all constraints i.e. checking the variance matrix produced is positive definite. If these conditions are not satisfied then the value is immediately rejected, otherwise we perform a standard Metropolis update and compare the log likelihoods with both the current and proposed values. As we are using Normal proposals rather than the truncated Normal proposals used in Chapter 9 we have symmetry and therefore do not need to calculate a Hastings ratio.

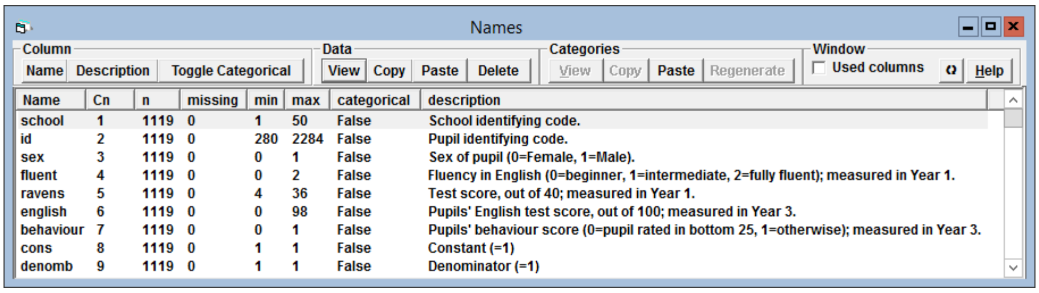
We will now consider an example from the field of education.

19.2 The JSP mixed response example

Our example dataset is a subset of the junior school project (JSP) dataset (Mortimore et al., 1988) and contains data on 1119 pupils from 47 schools. To view the data we need to load the worksheet and open the **Names** window as shown below.

- Select **Open Worksheet** from the **File** menu.
- Select **jspmix1.ws** from the list of worksheets.
- Click on the **Open** button.

Our response variables are for each pupil an English test score (**english**) marked out of 100 and a behaviour rating (**behaviour**), which is coded 0 if the pupil is rated in the bottom 25% and 1 otherwise. Note that this response originally had three categories but we have combined the top 25% and middle 50% categories so that we can fit the response using a Binomial distribution. Both these responses were measured at year 3 of the pupil's schooling.



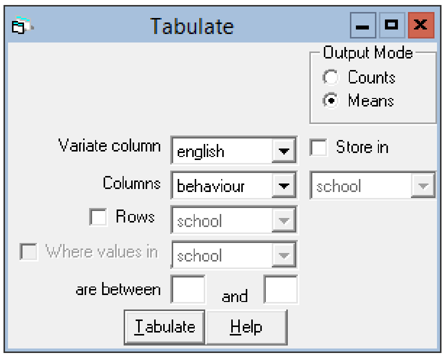
Name	Cn	n	missing	min	max	categorical	description
school	1	1119	0	1	50	False	School identifying code.
id	2	1119	0	280	2284	False	Pupil identifying code.
sex	3	1119	0	0	1	False	Sex of pupil (0=Female, 1=Male).
fluent	4	1119	0	0	2	False	Fluency in English (0=beginner, 1=intermediate, 2=fully fluent); measured in Year 1.
ravens	5	1119	0	4	36	False	Test score, out of 40; measured in Year 1.
english	6	1119	0	0	98	False	Pupils' English test score, out of 100; measured in Year 3.
behaviour	7	1119	0	0	1	False	Pupils' behaviour score (0=pupil rated in bottom 25, 1=otherwise); measured in Year 3.
cons	8	1119	0	1	1	False	Constant (=1)
denomb	9	1119	0	1	1	False	Denominator (=1)

We are interested in if there is a correlation between badly behaved pupils and poor exam marks, and we are also interested in the effect of predictor variables on both the exam scores and the behaviour of the pupils. Our potential predictor variables are gender (**sex** with 0 for a girl and 1 for a boy), a fluency in English indicator taken at year 1 (**fluent** with 0 for beginner, 1 for intermediate and 2 for fully fluent) and a test score at year 1 (**ravens** score out of 40).

We could look first at a simple tabulation between the response variables.

- Select **Tabulate** from the **Basic Statistics** menu.
- Select **Means** as **Output Mode**.
- Select **english** as the **Variate** column.
- Select **behaviour** as the **Columns** indicator.

The window will look as follows:



Output Mode: Counts [✓ Means]
Variate column: english
Columns: behaviour
☐ Store in: school
☐ Rows: school
☐ Where values in: school
are between ___ and ___

Now if we click on the **Tabulate** button we get the following results.

	0	1	TOTALS
N	248	871	1119
MEANS	28.9	45.1	41.5
SD'S	18.5	21.0	20.5

So here we see that the badly behaved pupils have on average a 16.2 points worse performance in English. We can look at this relationship further by investigating correlations between the two responses and additionally the three predictors:

- Select **Averages and Correlation** from the **Basic Statistics** menu.
- Click on **Correlation** in the **Operation** box.
- Select columns **sex**, **fluent**, **ravens**, **english** and **behaviour** (note you can use the Shift or Ctrl keys to do this).
- Click on the **Calculate** button.

This will execute the CORM command which gives means and sds for each variable along with the following correlation matrix:

Correlations

	sex	**fluent**	**ravens**	**english**	**behaviour**
sex	1.0000				
fluent	-0.0229	1.0000			
ravens	0.0341	0.1705	1.0000		
english	-0.1479	0.2054	0.5042	1.0000	
behaviour	-0.1272	-0.0038	0.2181	0.3122	1.0000

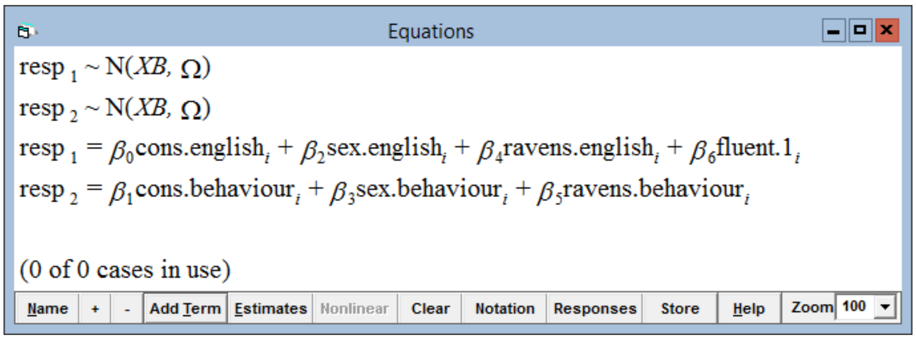
Here we see for **english**, positive correlations greater than 0.2 with both fluency and **ravens** and a negative correlation of approximately -0.15 with gender. For **behaviour** we see a positive correlation of greater than 0.2 with **ravens** and a negative correlation with gender but virtually no correlation with fluency. There is also a strong positive correlation (0.31) between the two responses. Based on this correlation matrix we can set up a single level model with fixed effects for each of the large correlations.

19.3 Setting up a single level mixed response model

We can start by setting up a multivariate Normal model via the **Equations** window as follows:

- Firstly we specify the responses by clicking on the **Responses** button.
- Select **english** and **behaviour** from the list that appears and click on the **Done** button.
- Click on the **Add Term** button and select **cons** from the variable list.
- Click on the **add Separate coefficients** button.
- Click on the **Add Term** button and select **sex** from the **variable** list.
- Click on the **add Separate coefficients** button.
- Click on the **Add Term** button and select **ravens** from the **variable** list.
- Click on the **add Separate coefficients** button.
- Click on the **Add Term** button and select **fluent** from the **variable** list.
- Click on the **add Common coefficient** button.
- On the screen that appears select **english** and click on the **Done** button.

Note that the variable **fluent** is not being fitted as a fixed effect for **behaviour** as there was no correlation between these two variables. The **Equations** window should now look as follows:



$\mathrm{resp}_1 \sim \mathrm{N}(XB, \Omega)$

$\mathrm{resp}_2 \sim \mathrm{N}(XB, \Omega)$

$\mathrm{resp}_1 = \beta_0\mathrm{cons.english}_i + \beta_2\mathrm{sex.english}_i + \beta_4\mathrm{ravens.english}_i + \beta_6\mathrm{fluent.1}_i$

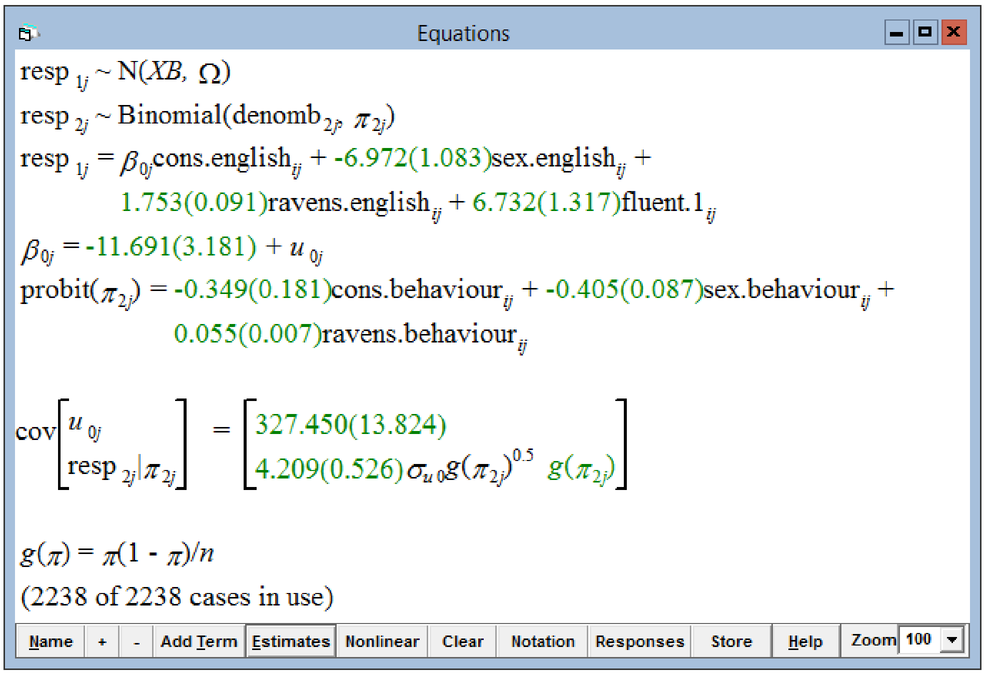
$\mathrm{resp}_2 = \beta_1\mathrm{cons.behaviour}_i + \beta_3\mathrm{sex.behaviour}_i + \beta_5\mathrm{ravens.behaviour}_i$

(0 of 0 cases in use)

We now need to specify the level 2 identifiers and to let the software know that the **behaviour** response should be treated as a Binomial response. To do this we need to specify the following in the **Equations** window:

- Click on the responses to bring up the **Y variable** window.
- Select **2-ij** as the number of levels and **id** as the level 2 identifier.
- Click on the **Done** button.
- Click on the **N** for response 2 and on the window that appears select **Binomial** for the distribution and **probit** for the link function.
- Click on the **Done** button.
- Click on the red n_{2j} that appears and select **denomb** as the denominator variable. Click on the **Done** button.
- Click on β_0 (**cons.english**) and select **j(id_long)**.
- Click on the **Done** button.
- Click on the **Nonlinear** button.
- On the **Nonlinear** window click on the **Use defaults** and the **Done** buttons.
- Click on the **Start** button.

This will have now set up the mixed response model and run it using the IGLS 1st order MQL method and the results should be as follows:



$\text{resp}_{1j} \sim \text{N}(XB, \Omega)$

$\text{resp}_{2j} \sim \text{Binomial}(\text{denomb}_{2j}, \pi_{2j})$

$\text{resp}_{1j} = \beta_{0j}\text{cons.english}_{ij} + -6.972(1.083)\text{sex.english}_{ij} + 1.753(0.091)\text{ravens.english}_{ij} + 6.732(1.317)\text{fluent.1}_{ij}$

$\beta_{0j} = -11.691(3.181) + u_{0j}$

$\text{probit}(\pi_{2j}) = -0.349(0.181)\text{cons.behaviour}_{ij} + -0.405(0.087)\text{sex.behaviour}_{ij} + 0.055(0.007)\text{ravens.behaviour}_{ij}$

$$\text{cov}\begin{bmatrix} u_{0j} \\ \text{resp}_{2j}|\pi_{2j} \end{bmatrix} = \begin{bmatrix} 327.450(13.824) & \\ 4.209(0.526)\,\sigma_{u0}g(\pi_{2j})^{0.5} & g(\pi_{2j}) \end{bmatrix}$$

$g(\pi) = \pi(1 - \pi)/n$

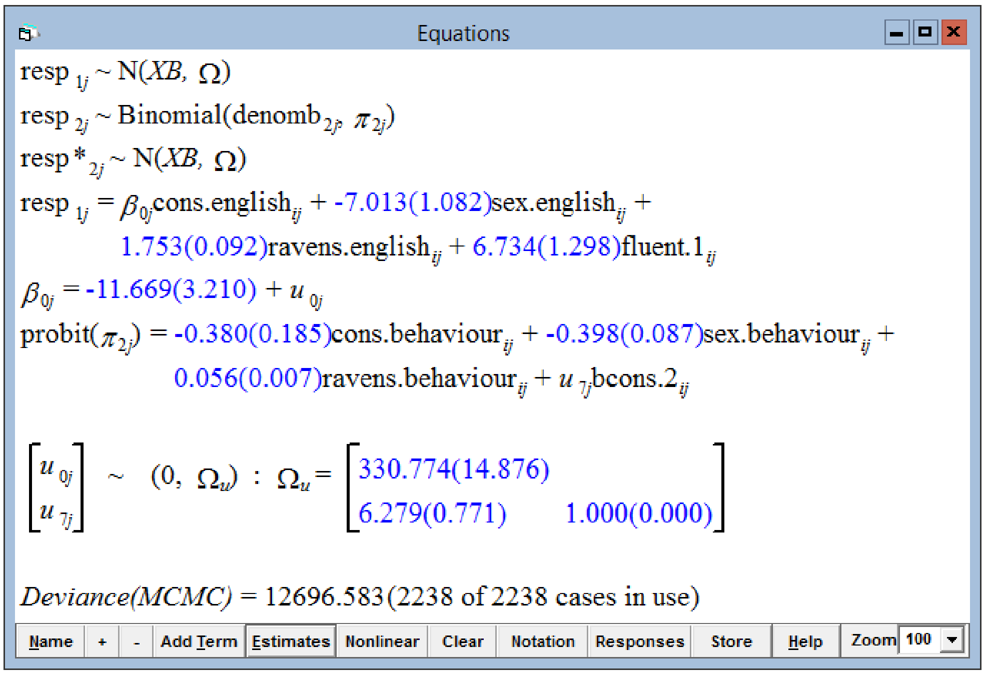
(2238 of 2238 cases in use)

Here we see that the individual level variance matrix contains some functions and this is due to the Taylor series approximations that are involved in the IGLS methods used here. The IGLS methods do not use the (underlying Normal) latent variable idea that is used by the MCMC method.

If we now want to run this model using MCMC we need to do the following:

- Change **Estimation method** to **MCMC**.
- Select **MCMC/MCMC Methods** from the **Model** menu.
- Click on **Reset** to select Gibbs sampling for fixed effects and residuals.
- Click on the **Done** button on the **Advanced MCMC Methodology Options** window.
- Click on the **Start** button.

After the 5,000 iterations have run we will get the following estimates:



$$\text{resp}_{1j} \sim \text{N}(XB, \Omega)$$
$$\text{resp}_{2j} \sim \text{Binomial}(\text{denomb}_{2j}, \pi_{2j})$$
$$\text{resp}^*_{2j} \sim \text{N}(XB, \Omega)$$
$$\text{resp}_{1j} = \beta_{0j}\text{cons.english}_{ij} + -7.013(1.082)\text{sex.english}_{ij} + 1.753(0.092)\text{ravens.english}_{ij} + 6.734(1.298)\text{fluent.1}_{ij}$$
$$\beta_{0j} = -11.669(3.210) + u_{0j}$$
$$\text{probit}(\pi_{2j}) = -0.380(0.185)\text{cons.behaviour}_{ij} + -0.398(0.087)\text{sex.behaviour}_{ij} + 0.056(0.007)\text{ravens.behaviour}_{ij} + u_{7j}\text{bcons.2}_{ij}$$
$$\begin{bmatrix} u_{0j} \\ u_{7j} \end{bmatrix} \sim (0,\ \Omega_u) \ :\ \Omega_u = \begin{bmatrix} 330.774(14.876) & \\ 6.279(0.771) & 1.000(0.000) \end{bmatrix}$$

Deviance(MCMC) = 12696.583(2238 of 2238 cases in use)

Here we see reasonably similar estimates to the IGLS method. The covariance parameter is however quite a bit larger (6.279 versus 4.209). This is because using MCMC this parameter represents the covariance between the **english** response and an unknown continuous **behaviour** response, which is positive when the observed dichotomous **behaviour** response equals one and negative when it equals zero. The IGLS method correlation is calculated differently. Here we see that the correlation between the pairs of residuals equals $0.345 = 6.279/\sqrt{330.774}$ suggesting that there is a strong positive correlation between English score and behaviour in the classroom even after accounting for other predictors.

We can also see the significant positive effects of fluency and Ravens test score on English marks and Ravens test score on behaviour and the negative effect of gender suggesting that girls both do better at English and are better behaved at year 3. It should here be noted that we cannot currently use the

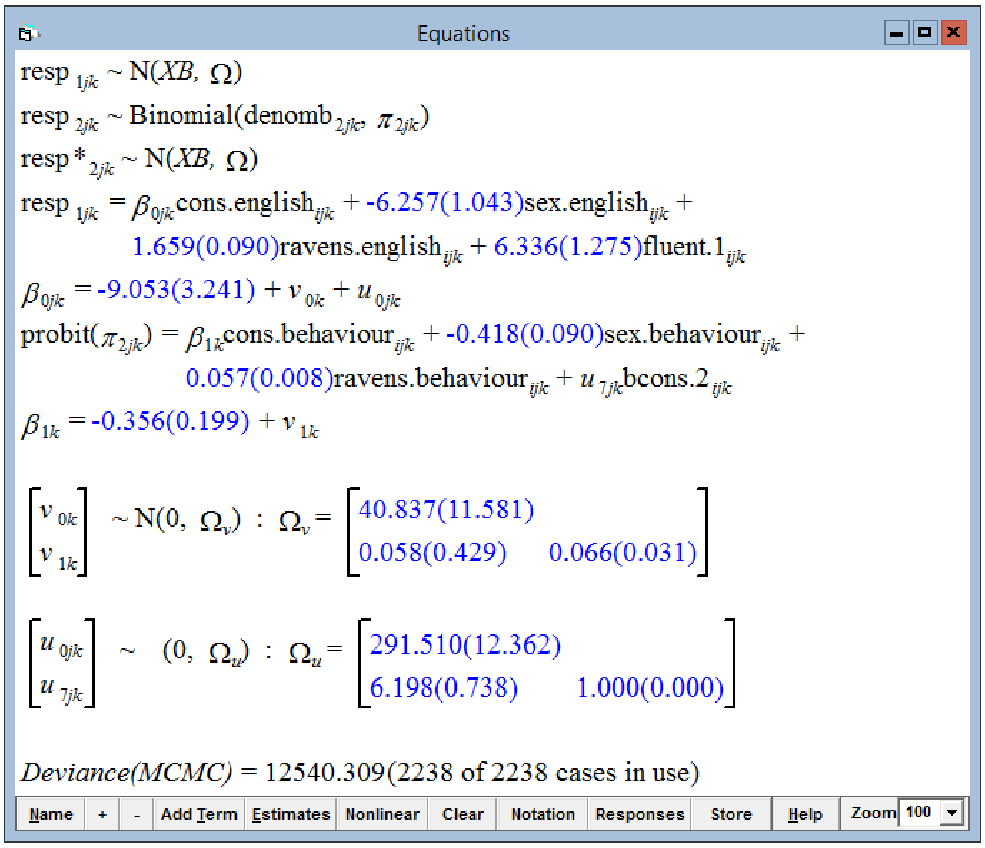
DIC diagnostic for these models as the deviance function is not worked out correctly.

19.4 Multilevel mixed response model

We can now extend our model to a multilevel model by adding in random effects for school for both responses as follows:

- Change **Estimation method** to **IGLS**.
- In the **Equations** window click on the response variables.
- Select **3-ijk** from the **N levels** list and **school** from the **level 3(k)** pull down list.
- Click on β_{0j} (**cons.english**) in the window and select the **k(school_long)** tickbox.
- Click on the **Done** button.
- Click on β_1 (**cons.behaviour**) in the window and select the **k(school_long)** tickbox.
- Click on the **Done** button.
- Click on the **Start** button (to get starting values from IGLS).
- Change **Estimation method** to **MCMC**.
- Click on the **Start** button.

Upon running for 5,000 iterations we get the following estimates:



$$\text{resp}_{1jk} \sim \text{N}(XB, \Omega)$$
$$\text{resp}_{2jk} \sim \text{Binomial}(\text{denomb}_{2jk}, \pi_{2jk})$$
$$\text{resp}^*_{2jk} \sim \text{N}(XB, \Omega)$$
$$\text{resp}_{1jk} = \beta_{0jk}\text{cons.english}_{ijk} + \text{-}6.257(1.043)\text{sex.english}_{ijk} + 1.659(0.090)\text{ravens.english}_{ijk} + 6.336(1.275)\text{fluent.1}_{ijk}$$
$$\beta_{0jk} = \text{-}9.053(3.241) + v_{0k} + u_{0jk}$$
$$\text{probit}(\pi_{2jk}) = \beta_{1k}\text{cons.behaviour}_{ijk} + \text{-}0.418(0.090)\text{sex.behaviour}_{ijk} + 0.057(0.008)\text{ravens.behaviour}_{ijk} + u_{7jk}\text{bcons.2}_{ijk}$$
$$\beta_{1k} = \text{-}0.356(0.199) + v_{1k}$$
$$\begin{bmatrix} v_{0k} \\ v_{1k} \end{bmatrix} \sim \text{N}(0, \Omega_v) : \Omega_v = \begin{bmatrix} 40.837(11.581) & \\ 0.058(0.429) & 0.066(0.031) \end{bmatrix}$$
$$\begin{bmatrix} u_{0jk} \\ u_{7jk} \end{bmatrix} \sim (0, \Omega_u) : \Omega_u = \begin{bmatrix} 291.510(12.362) & \\ 6.198(0.738) & 1.000(0.000) \end{bmatrix}$$

Deviance(MCMC) = 12540.309(2238 of 2238 cases in use)

Here we see that the positive correlation between responses persists at the individual level ($0.363 = 6.198/\sqrt{291.510 \times 1.000}$) but is negligible at the school level ($0.035 = 0.058/\sqrt{40.837 \times 0.066}$), and so schools with more disruptive children do not necessarily have worse average English attainment. All the fixed effects are little changed by adding in the school effects. School effects explain 12.3% (40.837/(291.510+40.837)) of English attainment residual variation and 6.2% (0.066/1.066) of behaviour score residual variation. Unfortunately due to the DIC diagnostic being unavailable for these models we cannot compare this model with the single level model, but the size of the coefficients appears to indicate a large improvement from introducing random effects.

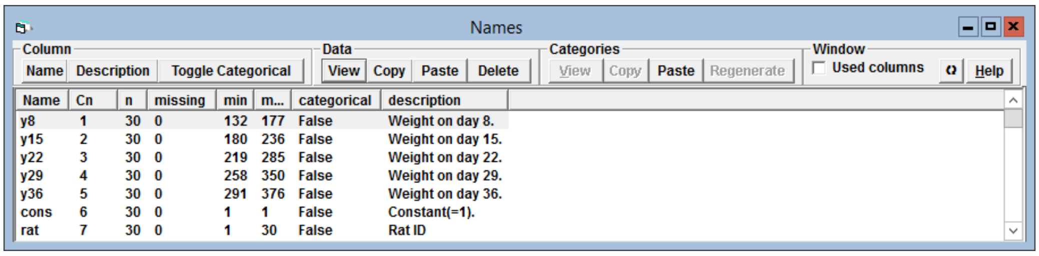
19.5 Rats dataset

Our second example of a model with a constrained variance matrix comes from an animal growth dataset discussed in Gelfand et al. (1990). The data consists of the weights of 30 rats measured weekly over a five week period. We will firstly load up the dataset and look at the **Names** window:

- Select **Open Sample Worksheet** from the **File** menu.
- Select **rats.ws** from the list of worksheets.

- Click on the **Open** button.

The data will then appear as follows:



Name	Cn	n	missing	min	m...	categorical	description
y8	1	30	0	132	177	False	Weight on day 8.
y15	2	30	0	180	236	False	Weight on day 15.
y22	3	30	0	219	285	False	Weight on day 22.
y29	4	30	0	258	350	False	Weight on day 29.
y36	5	30	0	291	376	False	Weight on day 36.
cons	6	30	0	1	1	False	Constant(=1).
rat	7	30	0	1	30	False	Rat ID

Here we see the five weights for each rat are recorded as five variables, **y8**, **y15**, **y22**, **y29** and **y36**. Gelfand et al. (1990) considered fitting a single response two level random slopes model with measurements nested within rats. We could repeat their analysis by stacking the data for the five responses into one column as illustrated, for an example from education, in Chapter 13 of the User's Guide to MLwiN. However here we will consider the data as five separate responses and fit a single level multivariate Normal response model.

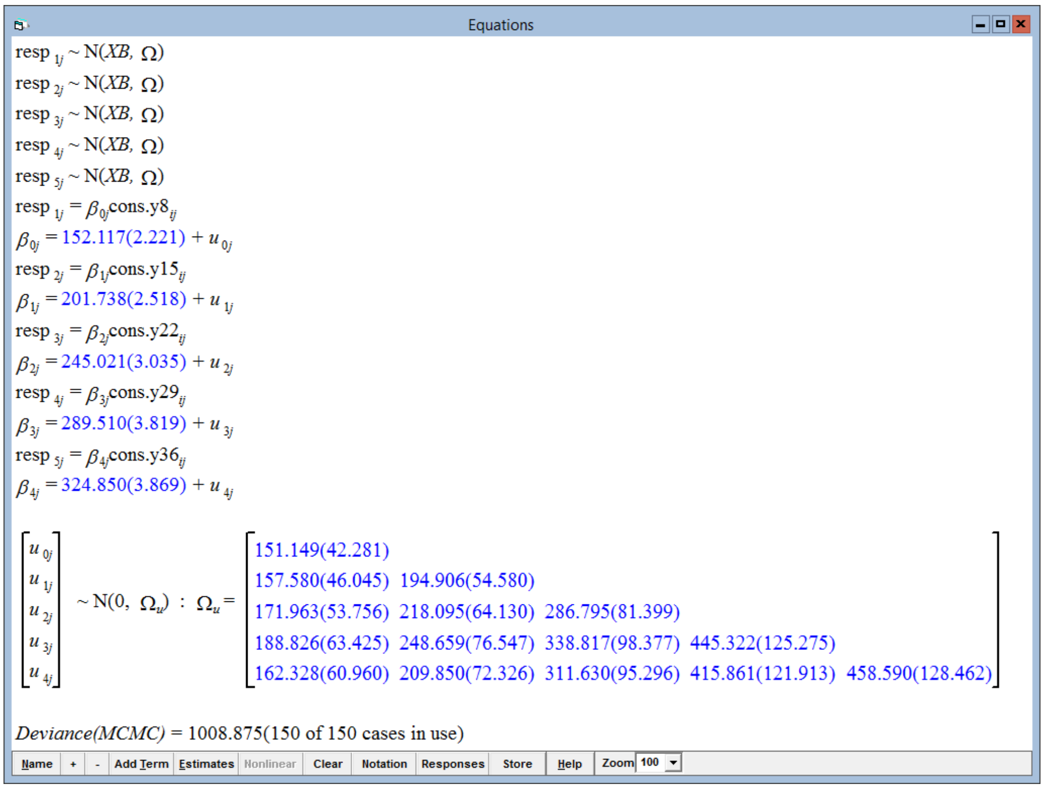
- Select **Equations** from the **Model** menu.
- Click on the **Responses** button to bring up the **Responses** window.
- Select **y8**, **y15**, **y22**, **y29** and **y36** as **y** variables.
- Click on the **Done** button.
- Click on the response names and select **2-ij** as the number of levels.
- Select **rat** as the **level 2(j)** identifier and click on the **Done** button.

We have now set up the responses and the level identifiers and we next need to add the predictors.

- Click on the **Add Term** button and select **cons** from the **variable** list.
- Click on the **add Separate coefficients** button on the **Add Term** window.
- Click on β_0 (**cons_y8**) and select the **j(rat_long)** tick box and click on the **Done** button.
- Click on β_1 (**cons_y15**) and select the **j(rat_long)** tick box and click on the **Done** button.
- Click on β_2 (**cons_y22**) and select the **j(rat_long)** tick box and click on the **Done** button.

- Click on β_3 (**cons_y29**) and select the **j(rat_long)** tick box and click on the **Done** button.
- Click on β_4 (**cons_y36**) and select the **j(rat_long)** tick box and click on the **Done** button.
- Click on the **Start** button.
- Change **Estimation method** to **MCMC**.
- Click on the **Start** button.

This will run the model using IGLS and then MCMC and the estimates after 5,000 iterations will be as follows:



Equations

$\text{resp}_{1j} \sim \text{N}(XB, \Omega)$
$\text{resp}_{2j} \sim \text{N}(XB, \Omega)$
$\text{resp}_{3j} \sim \text{N}(XB, \Omega)$
$\text{resp}_{4j} \sim \text{N}(XB, \Omega)$
$\text{resp}_{5j} \sim \text{N}(XB, \Omega)$
$\text{resp}_{1j} = \beta_{0j}\text{cons.y8}_{ij}$
$\beta_{0j} = 152.117(2.221) + u_{0j}$
$\text{resp}_{2j} = \beta_{1j}\text{cons.y15}_{ij}$
$\beta_{1j} = 201.738(2.518) + u_{1j}$
$\text{resp}_{3j} = \beta_{2j}\text{cons.y22}_{ij}$
$\beta_{2j} = 245.021(3.035) + u_{2j}$
$\text{resp}_{4j} = \beta_{3j}\text{cons.y29}_{ij}$
$\beta_{3j} = 289.510(3.819) + u_{3j}$
$\text{resp}_{5j} = \beta_{4j}\text{cons.y36}_{ij}$
$\beta_{4j} = 324.850(3.869) + u_{4j}$

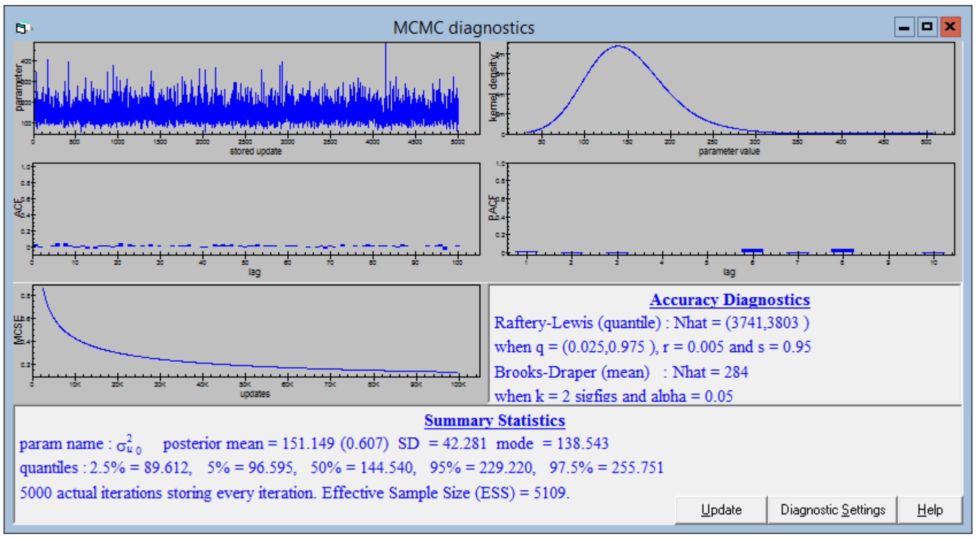
$$\begin{bmatrix} u_{0j} \\ u_{1j} \\ u_{2j} \\ u_{3j} \\ u_{4j} \end{bmatrix} \sim \text{N}(0, \Omega_u) : \Omega_u = \begin{bmatrix} 151.149(42.281) & & & & \\ 157.580(46.045) & 194.906(54.580) & & & \\ 171.963(53.756) & 218.095(64.130) & 286.795(81.399) & & \\ 188.826(63.425) & 248.659(76.547) & 338.817(98.377) & 445.322(125.275) & \\ 162.328(60.960) & 209.850(72.326) & 311.630(95.296) & 415.861(121.913) & 458.590(128.462) \end{bmatrix}$$

Deviance(MCMC) = 1008.875(150 of 150 cases in use)

Interestingly with such a small dataset and a large number of parameters there will be both a greater dependence on the prior distributions used, and the variance parameters will have skewed posterior distributions. For example if we look at the trace for the variance of the 8 day old measurements (σ^2_{u0}):

- Select the **Trajectories** window from the **Model** menu.
- Click on the trace plot for σ^2_{u0}.
- Click on **Yes** to calculate diagnostics.

The diagnostics for this parameter will then appear as follows:



Accuracy Diagnostics
Raftery-Lewis (quantile) : Nhat = (3741,3803)
when q = (0.025,0.975), r = 0.005 and s = 0.95
Brooks-Draper (mean) : Nhat = 284
when k = 2 sigfigs and alpha = 0.05

Summary Statistics
param name : σ^2_{u0} posterior mean = 151.149 (0.607) SD = 42.281 mode = 138.543
quantiles : 2.5% = 89.612, 5% = 96.595, 50% = 144.540, 95% = 229.220, 97.5% = 255.751
5000 actual iterations storing every iteration. Effective Sample Size (ESS) = 5109.

Here it is worth noting the differences between the mean, median and mode for this parameter due to the skewness. When we look at the DIC diagnostic available from the **Model** menu we get the following:

Dbar	D(thetabar)	pD	DIC
1008.87	993.29	15.58	1024.46

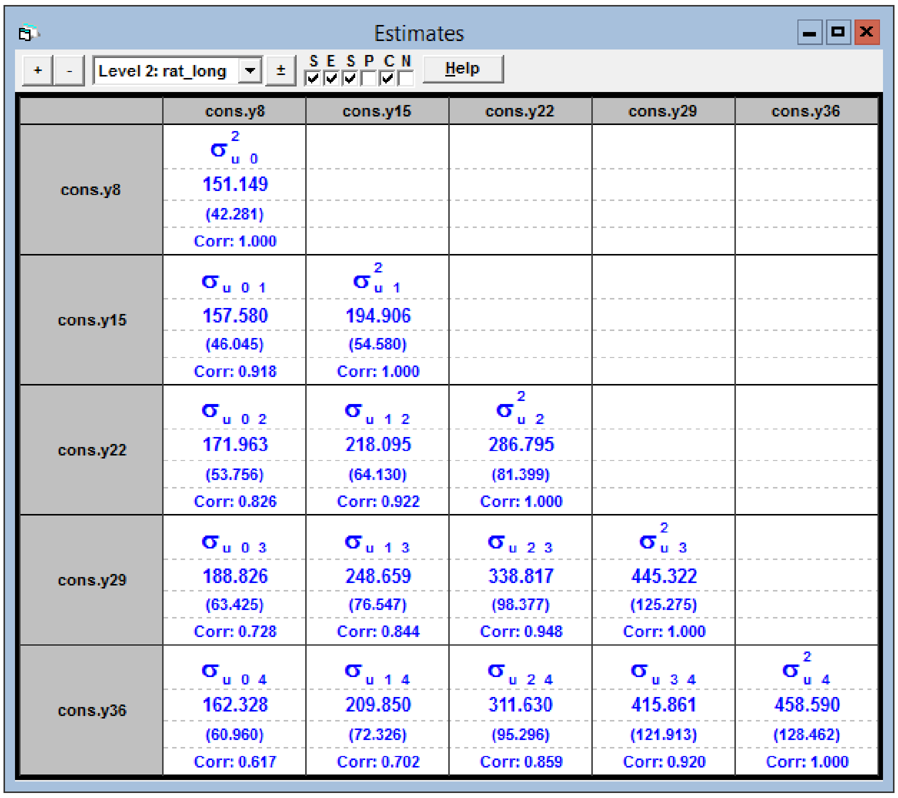
In fact this model has 20 parameters but the D(thetabar) statistic is sensitive to which plug-in estimates we use.

19.6 Fitting an autoregressive structure to the variance matrix

We could look at the variance in more detail and in particular look at the corresponding correlation matrix using the **Estimates** window.

- Select the **Estimate Tables** window from the **Model** menu.
- Click on the pull down list and choose **Level 2: rat_long**.
- Remove the tick from the **P** (past value) box.
- Add a tick to the **C** (correlations) box.

The window will then look as follows:



	cons.y8	cons.y15	cons.y22	cons.y29	cons.y36
cons.y8	σ^2_{u0} 151.149 (42.281) Corr: 1.000				
cons.y15	σ_{u01} 157.580 (46.045) Corr: 0.918	σ^2_{u1} 194.906 (54.580) Corr: 1.000			
cons.y22	σ_{u02} 171.963 (53.756) Corr: 0.826	σ_{u12} 218.095 (64.130) Corr: 0.922	σ^2_{u2} 286.795 (81.399) Corr: 1.000		
cons.y29	σ_{u03} 188.826 (63.425) Corr: 0.728	σ_{u13} 248.659 (76.547) Corr: 0.844	σ_{u23} 338.817 (98.377) Corr: 0.948	σ^2_{u3} 445.322 (125.275) Corr: 1.000	
cons.y36	σ_{u04} 162.328 (60.960) Corr: 0.617	σ_{u14} 209.850 (72.326) Corr: 0.702	σ_{u24} 311.630 (95.296) Corr: 0.859	σ_{u34} 415.861 (121.913) Corr: 0.920	σ^2_{u4} 458.590 (128.462) Corr: 1.000

Here if we look at the correlations closely we see that the correlation is high between consecutive observations but decreases as observations get further apart. One possible method to account for this and reduce the number of parameters used in the model is to impose a structure on these correlations. Given the observed values a natural candidate would be to fit an AR(1) correlation structure. Here the correlation between any two observations is equal to ρ to the power of the distance between the observations.

This can be done in MLwiN as follows:

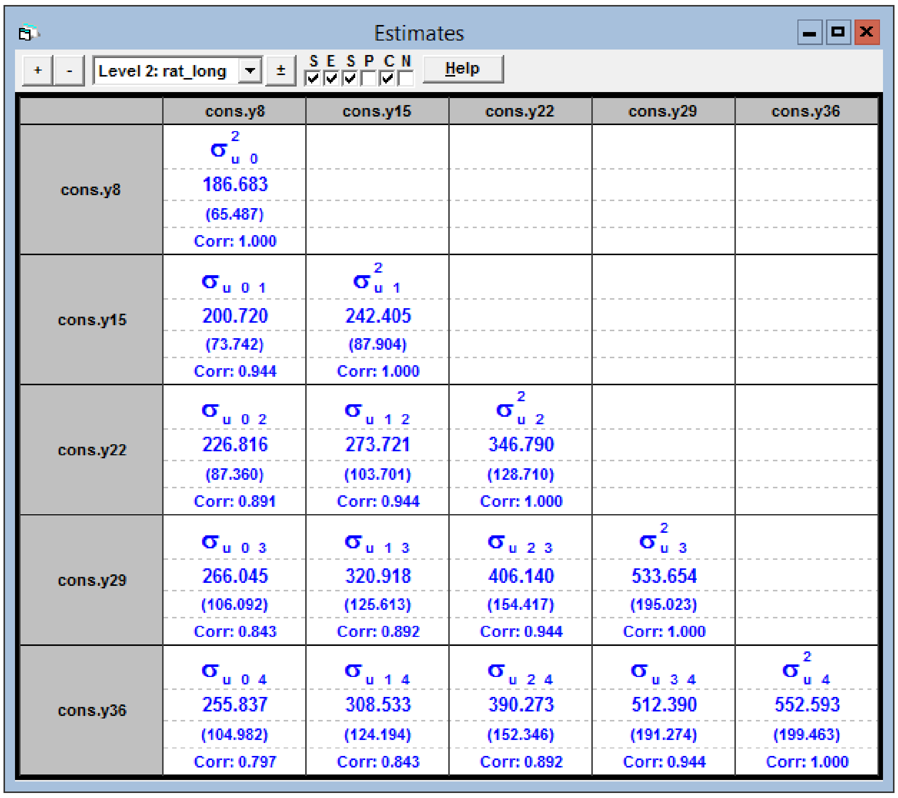
- Change **Estimation method** to **IGLS**.
- Click on the **Start** button.
- Change **Estimation method** to **MCMC**.
- Select **MCMC/Correlated Residuals** from the **Model** menu.
- Choose **AR1 structure/independent variances** as **Current Correlation structure**.
- Click on the **Done** button.

The options to change the structure of the variance matrix are fairly limited. They only work on the bottom level variance matrix of a multivariate model and there are currently four special cases that are dealt with. These consist of either all equal correlations or AR1 correlation structures combined with either independent or all equal variances.

For all these four combinations we use univariate Metropolis sampling for the variance elements of the lowest level variance matrix and the correlation parameter ρ and then check that the variance matrix formed is positive definite. All other steps of the algorithm are unchanged and use Gibbs sampling. As the dataset is small and we are running Metropolis steps we will run for 50,000 iterations.

- Select the **Estimation Control** window.
- Change the **Monitoring Chain Length** to 50,000.
- Change the refresh rate to every 500 iterations.
- Click on the **Start** button.

After the iterations have finished the **Estimates** window should now look as follows: (Note if you closed this window you will have to reopen it and select the options described earlier in the chapter.)



	cons.y8	cons.y15	cons.y22	cons.y29	cons.y36
cons.y8	σ^2_{u0} 186.683 (65.487) Corr: 1.000				
cons.y15	σ_{u01} 200.720 (73.742) Corr: 0.944	σ^2_{u1} 242.405 (87.904) Corr: 1.000			
cons.y22	σ_{u02} 226.816 (87.360) Corr: 0.891	σ_{u12} 273.721 (103.701) Corr: 0.944	σ^2_{u2} 346.790 (128.710) Corr: 1.000		
cons.y29	σ_{u03} 266.045 (106.092) Corr: 0.843	σ_{u13} 320.918 (125.613) Corr: 0.892	σ_{u23} 406.140 (154.417) Corr: 0.944	σ^2_{u3} 533.654 (195.023) Corr: 1.000	
cons.y36	σ_{u04} 255.837 (104.982) Corr: 0.797	σ_{u14} 308.533 (124.194) Corr: 0.843	σ_{u24} 390.273 (152.346) Corr: 0.892	σ_{u34} 512.390 (191.274) Corr: 0.944	σ^2_{u4} 552.593 (199.463) Corr: 1.000

It should be noted here that when constraints are imposed on a variance matrix we are forced to use uniform priors on the individual elements of the variance matrix. In this case we have used uniform priors on the 5 variances and the correlation parameter ρ whereas previously we used an inverse Wishart prior. As can be seen the variance estimates under a Uniform prior are much larger. The interested reader should look at Browne & Draper (2000, 2006) for more information on choosing "uninformative" prior distributions.

If we look at the DIC diagnostic for this model we get:

Dbar	D(thetabar)	pD	DIC	
1015.65	1003.30	9.35	1025.00	(with constraint)
1008.87	993.29	15.58	1024.46	(without constraint)

Again the pD estimate is an underestimate of the true number of parameters which in this model is 11. As we can see from comparing DIC values, the reduction in parameters is balanced by an increase in the deviance from the full covariance model. This may partly be due to the different prior distributions but could also be due to the fact that the correlations decline more quickly than an AR1 process.

Chapter learning outcomes

- ⋆ How to fit models to mixtures of binary and continuous outcomes.
- ⋆ How to extend such models to the multilevel case.
- ⋆ How to fit patterned variance matrices in repeated measures modelling.

Chapter 20

Multilevel Factor Analysis Modelling

In the last two chapters we have looked at datasets where there is more than one response variable of interest. There are many common techniques used to model and summarise multivariate datasets (see Chatfield & Collins, 1980; Krzanowski & Marriott, 1994, 1995, for reviews). When we consider a multivariate response vector we are typically interested in explaining both the variation in the individual responses (as in single response modelling) and the correlation between the responses for individuals. As the number of responses increases so the number of correlations increases and we are often interested in looking at methods of describing these correlations with fewer parameters, for example by assuming an autocorrelation pattern as shown in the last chapter.

Often however even though we collect many responses on an individual, interest lies in finding differences between individuals (or groups of individuals) along a small number of summary dimensions rather than explaining differences in particular responses. Variable reduction techniques are used to find a smaller set of derived responses, which capture the majority of the variation in the dataset. Factor analysis is an example of a technique that attempts to do this.

20.1 Factor analysis modelling

In factor analysis modelling we aim to reduce the dimensionality of our response vector by creating 'common factors' as follows for a single level model:

$$y_{ir} = \sum_{j=1}^{J} \lambda_{rj}\eta_{ij} + e_{ir}$$

Here y_{ir} is the rth response for the ith individual, η_{ij} is the jth factor for the ith individual and λ_{rj} is the coefficient (known as the factor loading) for the rth response for factor j. The e_{ir} is the residual (sometimes called specific factor or uniqueness) for the rth response for the ith individual. To complete the model we give distributional assumptions for both the residuals and the factors as follows:

$$e_{ir} \sim N(0, \sigma^2_{er}), \eta_i \sim MVN(0, \Sigma_\eta)$$

Often the factors will be assumed independent and Σ_η will be diagonal with elements $\sigma^2_{\eta j}$ but we can allow correlated factors (see later). To make this model identifiable we need to either fix/constrain some of the factor loadings or the factor variances. If we wish to fit such a factor model in a Bayesian framework (as we will here) we also need to incorporate prior distributions for the factor loadings and all variances into the model. As for the multilevel models we have dealt with earlier we will assume 'improper' uniform priors for all loadings that are not fixed and $\Gamma^{-1}(10^{-3}, 10^{-3})$ priors (by default) for all variances.

20.2 MCMC algorithm

The MCMC algorithm for the factor model is very similar to the algorithm for a simple two level multilevel model. If we consider the factors as known values then the problem becomes several one-level regression models with the factor loadings as fixed effects in the regressions, which we can easily fit. If on the other hand we consider the loadings as fixed then the factors become level 2 residuals in a two level model, which we also know how to fit. Therefore we see that all parameters have simple known forms and so we can use a Gibbs sampling algorithm to fit the model.

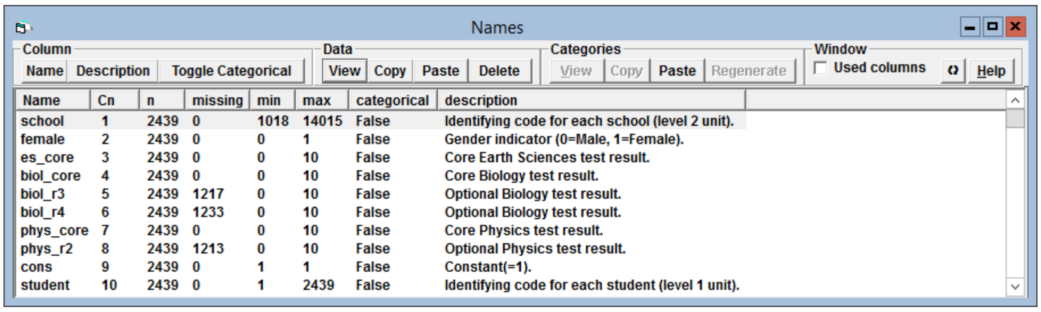
The great advantage of the Gibbs sampling algorithm is that we can of course now consider incorporating the factor analysis structure as a part of a more general multivariate multilevel model. This allows us to include fixed effects and higher-level structures in our model. Goldstein & Browne (2002) give general MCMC algorithms for this and we will see how this works later. Firstly however we will return to our Hungarian exam results example that we discussed in Chapter 18.

20.3 Hungarian science exam dataset

The dataset as described in Chapter 18 consists of exam results from six tests taken by 2,439 students in 1984. Of the tests all students took the three core papers and then one or two of the optional papers. In Chapter 18 we considered this dataset from a missing data viewpoint and used MCMC algorithms to impute the missing data. Here we will simply incorporate the missing data steps along with the factor analysis steps in our general MCMC algorithm.

- Select **Open Sample Worksheet** from the **File** menu.
- Select **hungary1.ws** from the list and click on the **Open** button.

The variables in the dataset are as follows:

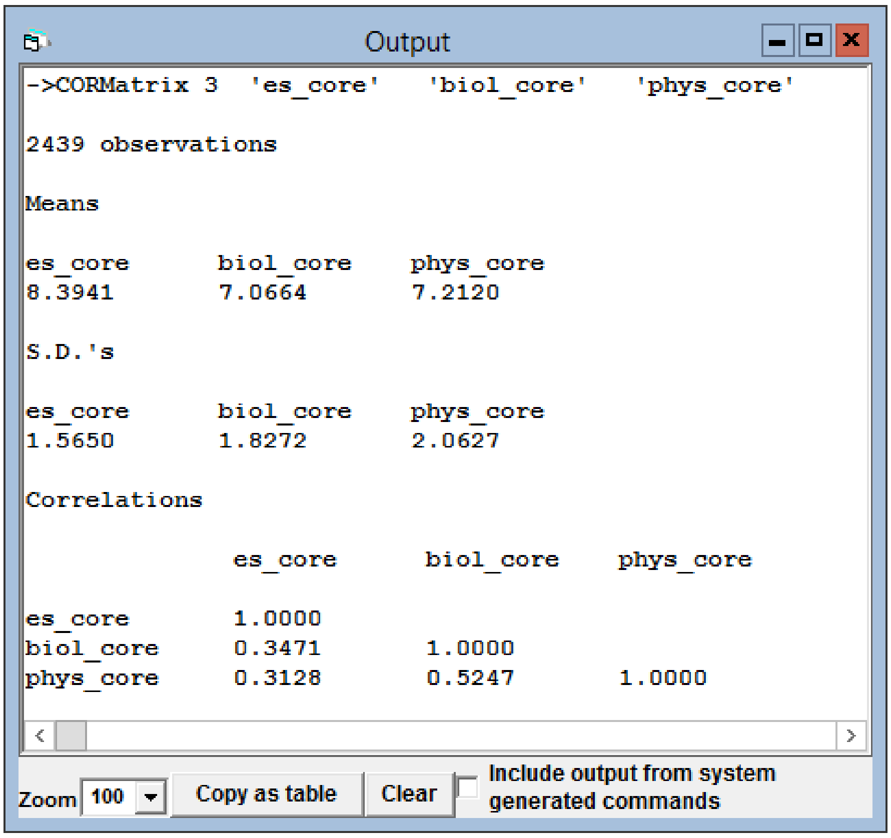


Name	Cn	n	missing	min	max	categorical	description
school	1	2439	0	1018	14015	False	Identifying code for each school (level 2 unit).
female	2	2439	0	0	1	False	Gender indicator (0=Male, 1=Female).
es_core	3	2439	0	0	10	False	Core Earth Sciences test result.
biol_core	4	2439	0	0	10	False	Core Biology test result.
biol_r3	5	2439	1217	0	10	False	Optional Biology test result.
biol_r4	6	2439	1233	0	10	False	Optional Biology test result.
phys_core	7	2439	0	0	10	False	Core Physics test result.
phys_r2	8	2439	1213	0	10	False	Optional Physics test result.
cons	9	2439	0	1	1	False	Constant(=1).
student	10	2439	0	1	2439	False	Identifying code for each student (level 1 unit).

Here as before we have two level identifiers, **school** and **student**, and the six test scores are stored in columns **c3–c8**. We will firstly look at the correlation matrix for the six responses.

- Select **Averages and Correlations** from the **Basic Statistics** menu.
- Click on the **Correlation** button.
- Select **es_core**, **biol_core**, and **phys_core** from the pull-down list (you can use the mouse and the Ctrl button to do this).

If we now click on the **Calculate** button we will see the following estimates:



```
->CORMatrix 3  'es_core'   'biol_core'   'phys_core'

2439 observations

Means

es_core       biol_core     phys_core
8.3941        7.0664        7.2120

S.D.'s

es_core       biol_core     phys_core
1.5650        1.8272        2.0627

Correlations

              es_core       biol_core     phys_core

es_core       1.0000
biol_core     0.3471        1.0000
phys_core     0.3128        0.5247        1.0000
```

Here we see that the correlations between the three core scores are all reasonably large, in particular between biology and physics scores. The **Correlations** option in MLwiN gives correlations based on complete responses only, and so as no individual has taken all six tests we cannot directly calculate a 6×6 correlation matrix. Instead we can set up a simple multivariate model that consists of only an intercept for each response and a full covariance matrix between the 6 responses.

- Select **Equations** from the **Model** menu.
- Click on the **Responses** button.
- Select **es_core**, **biol_core**, **biol_r3**, **biol_r4**, **phys_core** and **phys_r2** from the response list.
- Click on the **Done** button.
- Click on the responses and the **Y variable** window will appear.
- Select **2-ij** for the number of levels.
- Select **student** from the **level 2(j)** pull down list.
- Click on the **Done** button.

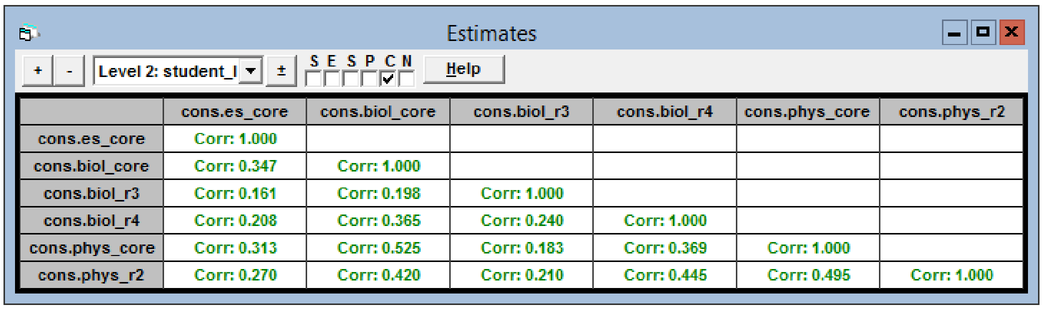
We have now defined both the response and level identifiers. We now need to add the predictors and random effects to the model.

- Click on the **Add Term** button.
- Select **cons** from the **variable** list on the **Add Term** window.
- Click on the **add Separate coefficients** button.
- Click on β_0 (**cons.es_core**) in the **Equations** window and select the **j(student_long)** tick box keeping **Fixed Parameter** selected.
- Click on the **Done** button
- Repeat this for the other 5 variables, β_1, β_2, β_3, β_4, and β_5.
- Click on the **Start** button to run the model using IGLS.

After the IGLS model converges we can look at the correlation matrix via the **Estimate Tables** window:

- Select **Estimate Tables** from the **Model** menu.
- Replace **Fixed Part** with **Level 2: student_long** in the pull down list.
- Remove the tick under S, E, S and P by clicking in the boxes and add a tick under the C.

The correlation matrix appears as follows:



	cons.es_core	cons.biol_core	cons.biol_r3	cons.biol_r4	cons.phys_core	cons.phys_r2
cons.es_core	Corr: 1.000					
cons.biol_core	Corr: 0.347	Corr: 1.000				
cons.biol_r3	Corr: 0.161	Corr: 0.198	Corr: 1.000			
cons.biol_r4	Corr: 0.208	Corr: 0.365	Corr: 0.240	Corr: 1.000		
cons.phys_core	Corr: 0.313	Corr: 0.525	Corr: 0.183	Corr: 0.369	Corr: 1.000	
cons.phys_r2	Corr: 0.270	Corr: 0.420	Corr: 0.210	Corr: 0.445	Corr: 0.495	Corr: 1.000

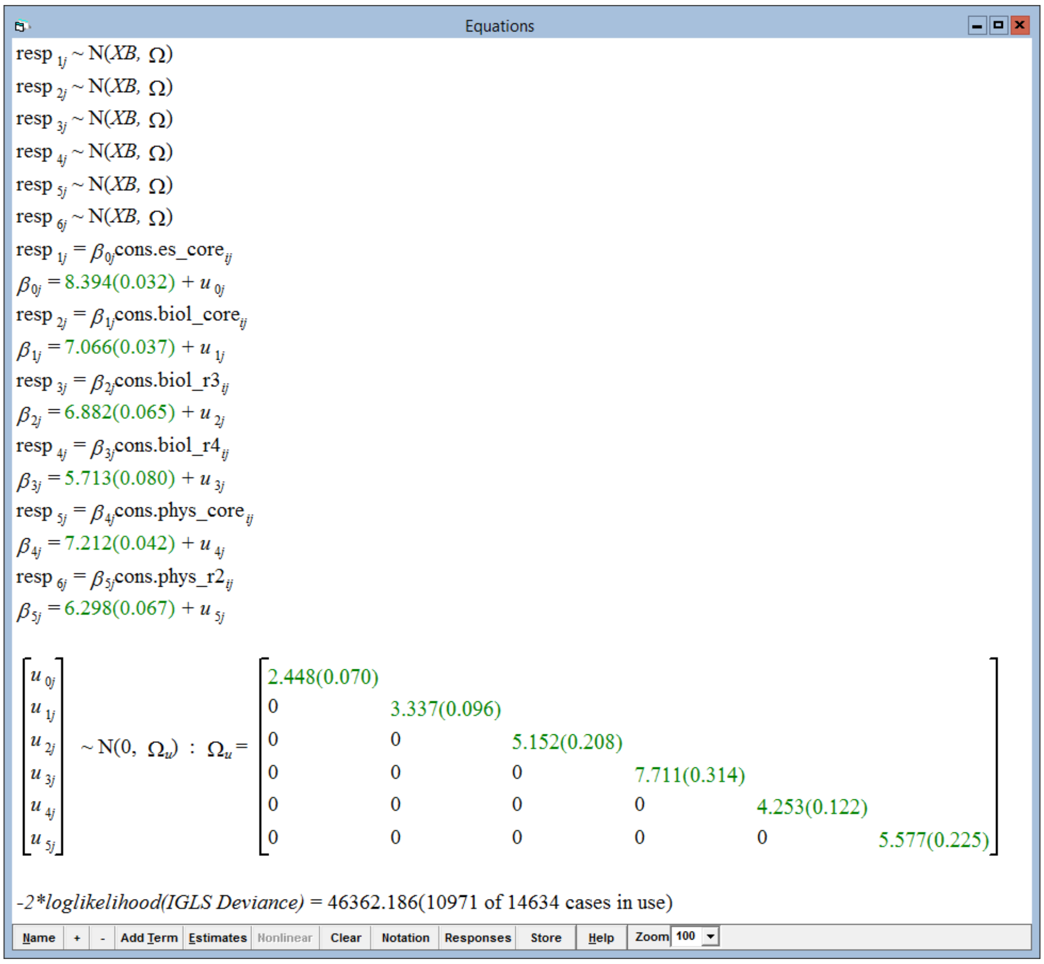
Here we see that the correlations between the core subjects are almost identical to the values obtained from the **Averages and Correlations** window. In fact if we were to select the responses in pairs via the **Averages and Correlations** window we will get similar estimates to those obtained here for each correlation (using all the individuals that have both responses). If we look at this matrix in more detail we see that all correlations are positive suggesting that students that do well in one paper will also generally do well in the other papers. The correlations range from 0.161 between earth sciences core and the optional r3 paper in biology to a correlation of 0.525 between the core papers in biology and physics. We will now try to explain these 15 correlations by using factor analysis.

20.4 A single factor Bayesian model

We will firstly consider fitting a single factor model. To do this we first need to remove all the covariances from the model to have independence between the responses. This is because these covariances will be explained by the factor variables that we add to the model.

- Select **Equations** from the **Model** menu.
- In the **Equations** window click on the **Estimates** button until the estimates appear as green numbers.
- Scroll down until Ω_u is visible.
- Click on the Ω_u and select **set diagonal matrix**.
- When finished click on the **Start** button.

We will have now run a model that consists of 6 intercepts with independent random effects, one for each response. The **Equations** window should look as follows:



$\text{resp}_{1j} \sim \text{N}(XB,\ \Omega)$
$\text{resp}_{2j} \sim \text{N}(XB,\ \Omega)$
$\text{resp}_{3j} \sim \text{N}(XB,\ \Omega)$
$\text{resp}_{4j} \sim \text{N}(XB,\ \Omega)$
$\text{resp}_{5j} \sim \text{N}(XB,\ \Omega)$
$\text{resp}_{6j} \sim \text{N}(XB,\ \Omega)$
$\text{resp}_{1j} = \beta_{0j}\text{cons.es\_core}_{ij}$
$\beta_{0j} = 8.394(0.032) + u_{0j}$
$\text{resp}_{2j} = \beta_{1j}\text{cons.biol\_core}_{ij}$
$\beta_{1j} = 7.066(0.037) + u_{1j}$
$\text{resp}_{3j} = \beta_{2j}\text{cons.biol\_r3}_{ij}$
$\beta_{2j} = 6.882(0.065) + u_{2j}$
$\text{resp}_{4j} = \beta_{3j}\text{cons.biol\_r4}_{ij}$
$\beta_{3j} = 5.713(0.080) + u_{3j}$
$\text{resp}_{5j} = \beta_{4j}\text{cons.phys\_core}_{ij}$
$\beta_{4j} = 7.212(0.042) + u_{4j}$
$\text{resp}_{6j} = \beta_{5j}\text{cons.phys\_r2}_{ij}$
$\beta_{5j} = 6.298(0.067) + u_{5j}$

$$\begin{bmatrix} u_{0j} \\ u_{1j} \\ u_{2j} \\ u_{3j} \\ u_{4j} \\ u_{5j} \end{bmatrix} \sim \text{N}(0,\ \Omega_u)\ :\ \Omega_u = \begin{bmatrix} 2.448(0.070) & & & & & \\ 0 & 3.337(0.096) & & & & \\ 0 & 0 & 5.152(0.208) & & & \\ 0 & 0 & 0 & 7.711(0.314) & & \\ 0 & 0 & 0 & 0 & 4.253(0.122) & \\ 0 & 0 & 0 & 0 & 0 & 5.577(0.225) \end{bmatrix}$$

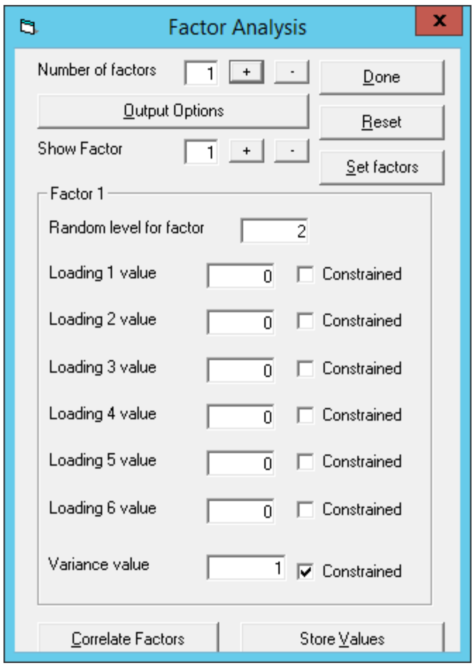
*-2*loglikelihood(IGLS Deviance)* = 46362.186(10971 of 14634 cases in use)

Name + - Add Term Estimates Nonlinear Clear Notation Responses Store Help Zoom 100

We now need to define the factor model.

- Change **Estimation method** to **MCMC**.
- Select **MCMC/Factor Analysis** from the **Model** menu.
- Click on the + button to increase the number of factors to 1.

The **Factor Analysis** screen will now appear as follows:



Factor Analysis

Number of factors	1	
Show Factor	1	
Random level for factor	2	
Loading 1 value	0	Constrained [ ]
Loading 2 value	0	Constrained [ ]
Loading 3 value	0	Constrained [ ]
Loading 4 value	0	Constrained [ ]
Loading 5 value	0	Constrained [ ]
Loading 6 value	0	Constrained [ ]
Variance value	1	Constrained [✓]

In factor analysis modelling we must be careful that we include enough constraints to ensure a unique identifiable solution. This is even more important when we use MCMC, as we have to avoid allowing the chains to move between rotational solutions. For example if a factor is multiplied by -1, and the respective loadings are also multiplied by -1 this will give an alternative solution that also fits the data. Such solutions are equivalent and a maximising algorithm will find just one of them whilst a simulation based method may move between the two and never converge.

Consequently in this chapter we will constrain the ith factor loading of the ith factor to equal 1 and constrain the jth factor loading of the ith factor to equal 0 $\forall j < i$. Geweke & Zhou (1996) give a similar (but not identical and slightly less rigid) system of constraints.

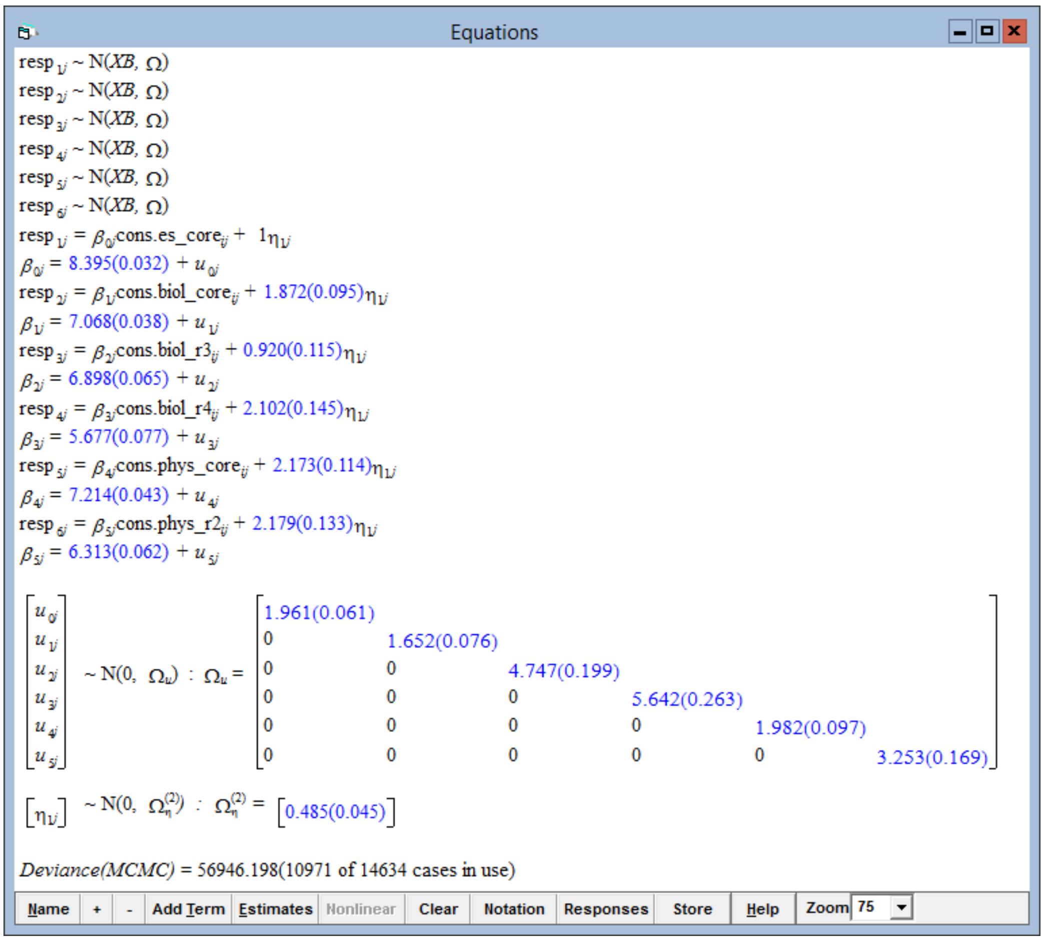
To use this set of constraints we need to do the following:

- Change the **Loading 1 value** to 1 and click in the **constraint** box for loading 1.
- Click in the **constraint** box for the **Variance value** to remove this constraint.

- Click on the **Set factors** button to set the factors.
- Click on the **Done** button.

This will add the factor to the model. We can now run this model by clicking on the **Start** button. It should be noted at this point that factor analysis models contain larger numbers of parameters than most of the models we have dealt with so far and so this first model will take several minutes to run.

After the estimation finishes we will get the following estimates:



Equations

$\text{resp}_{1j} \sim \text{N}(XB, \Omega)$
$\text{resp}_{2j} \sim \text{N}(XB, \Omega)$
$\text{resp}_{3j} \sim \text{N}(XB, \Omega)$
$\text{resp}_{4j} \sim \text{N}(XB, \Omega)$
$\text{resp}_{5j} \sim \text{N}(XB, \Omega)$
$\text{resp}_{6j} \sim \text{N}(XB, \Omega)$
$\text{resp}_{1j} = \beta_{0j}\text{cons.es\_core}_{ij} + 1\eta_{1j}$
$\beta_{0j} = 8.395(0.032) + u_{0j}$
$\text{resp}_{2j} = \beta_{1j}\text{cons.biol\_core}_{ij} + 1.872(0.095)\eta_{1j}$
$\beta_{1j} = 7.068(0.038) + u_{1j}$
$\text{resp}_{3j} = \beta_{2j}\text{cons.biol\_r3}_{ij} + 0.920(0.115)\eta_{1j}$
$\beta_{2j} = 6.898(0.065) + u_{2j}$
$\text{resp}_{4j} = \beta_{3j}\text{cons.biol\_r4}_{ij} + 2.102(0.145)\eta_{1j}$
$\beta_{3j} = 5.677(0.077) + u_{3j}$
$\text{resp}_{5j} = \beta_{4j}\text{cons.phys\_core}_{ij} + 2.173(0.114)\eta_{1j}$
$\beta_{4j} = 7.214(0.043) + u_{4j}$
$\text{resp}_{6j} = \beta_{5j}\text{cons.phys\_r2}_{ij} + 2.179(0.133)\eta_{1j}$
$\beta_{5j} = 6.313(0.062) + u_{5j}$

$$\begin{bmatrix} u_{0j} \\ u_{1j} \\ u_{2j} \\ u_{3j} \\ u_{4j} \\ u_{5j} \end{bmatrix} \sim \text{N}(0,\ \Omega_u) \ :\ \Omega_u = \begin{bmatrix} 1.961(0.061) & & & & & \\ 0 & 1.652(0.076) & & & & \\ 0 & 0 & 4.747(0.199) & & & \\ 0 & 0 & 0 & 5.642(0.263) & & \\ 0 & 0 & 0 & 0 & 1.982(0.097) & \\ 0 & 0 & 0 & 0 & 0 & 3.253(0.169) \end{bmatrix}$$

$$\begin{bmatrix} \eta_{1j} \end{bmatrix} \sim \text{N}(0,\ \Omega_\eta^{(2)}) \ :\ \Omega_\eta^{(2)} = \begin{bmatrix} 0.485(0.045) \end{bmatrix}$$

Deviance(MCMC) = 56946.198(10971 of 14634 cases in use)

We see that the fixed effect parameters are roughly the same estimates as the model without a factor. Our interest lies in the estimates of the factor loadings and the factor values or ‘scores’ themselves.

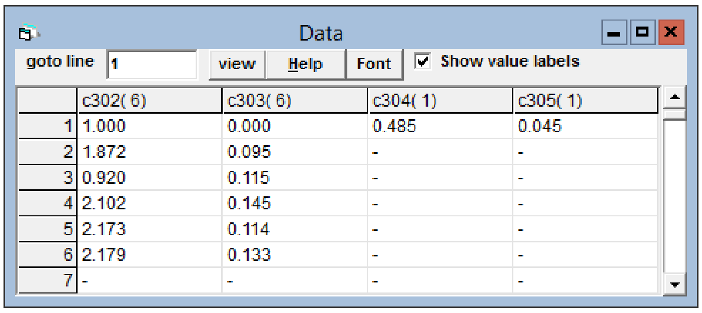
- Select **MCMC/Factor Analysis** from the **Model** menu.
- Click on the **Output Options** button.
- The **Factor Analysis Output Options** window will appear.
- On this window click on the three tick boxes to output factor values, loadings and variances respectively.

- This will put estimates and standard errors in the default columns, **c300-c305**.
- Click on the **Calculate** button.
- Click on the **Done** buttons on both the **Factor output** and **Factor** windows.

By default for factor analysis models, estimates and standard errors are calculated by the MCMC engine (as with residuals in other models). As we will see later we can also store all or any of the chains for the parameters. We will now look at the estimates from the model.

- Select **View/Edit Data** from the **Data Manipulation** menu.
- Click on the **view** button and select columns **c302**, **c303**, **c304** and **c305** (note that you can use the Ctrl key to select multiple columns).
- Click on the **OK** button.

The data will look as follows:



	c302(6)	c303(6)	c304(1)	c305(1)
1	1.000	0.000	0.485	0.045
2	1.872	0.095	-	-
3	0.920	0.115	-	-
4	2.102	0.145	-	-
5	2.173	0.114	-	-
6	2.179	0.133	-	-
7	-	-	-	-

Here we see the factor loadings in column **c302** and the factor variance in **c304**. Note that these estimates are now also shown in the **Equations** window. The loadings are all positive and given that we saw all positive correlations between the responses we can think of this first factor as representing a measure of overall exam success. In fact if the factor variance was 1 then the loadings would represent the correlations between the factor and the responses. All these loadings are much greater than their standard errors (given in **c303**), which also gives our interpretation more support.

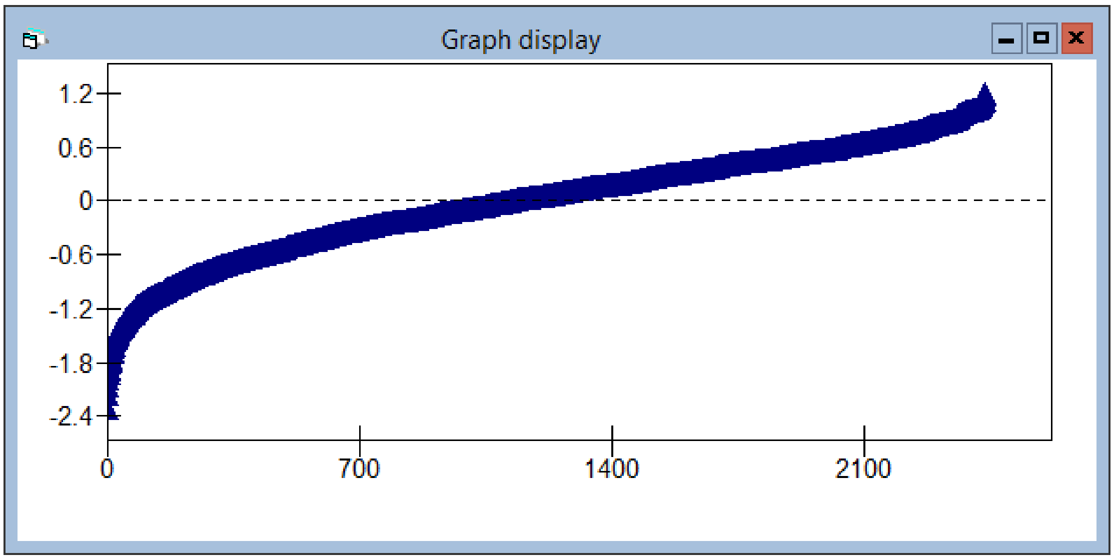
The actual factor scores for this model can be seen in column **c300**. We can rank these and plot them against their ranks as follows:

- Select **Command Interface** from the **Data Manipulation** menu and enter the command:

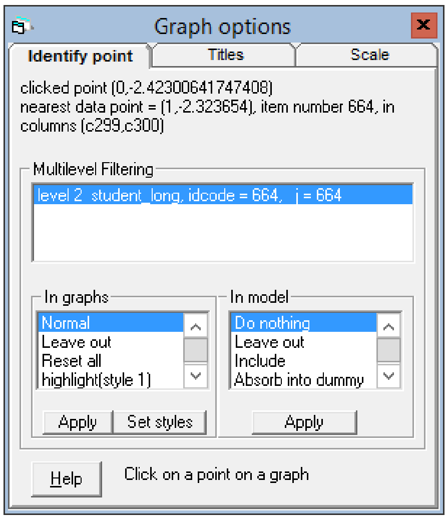
```
▶ RANK c300 c299
```

- Select **Customised Graphs** from the **Graphs** menu.
- Select **c300** as the y variable.
- Select **c299** as the x variable.
- Click on the **Apply** button.

The graph of the factor values will then be plotted as follows:



If we now click on the extreme points in the graph, for example the first (smallest) point, we will get the following window:



clicked point (0,-2.42300641747408)
nearest data point = (1,-2.323654), item number 664, in columns (c299,c300)

level 2 student_long, idcode = 664, j = 664

Click on a point on a graph

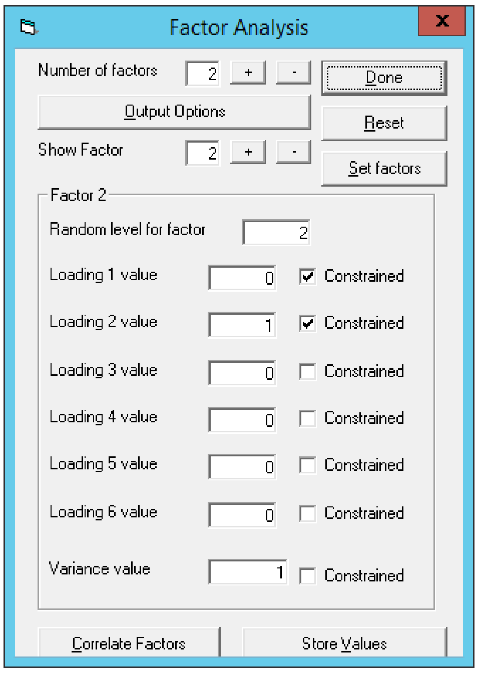
Here we see that the individual with the lowest factor value is student 664. Examining the data we see that this student got marks of 0 in both physics and biology core papers, 1.6667 in the earth sciences core paper and 5.7143 in the physics r2 additional paper. Similarly the individual with the highest factor value is student 2132. Examining their responses we see that they got full marks (10) in all the 5 papers they took. This backs up our inference that this factor is giving a measure of overall performance. The non-symmetry of the graph is due to the average marks in all papers being high and consequently there being greater ceiling effects than floor effects in the responses.

20.5 Adding a second factor to the model

The last model attempted to reduce the six response multivariate Normal model to a model with the dependence between the responses explained by one factor, which we interpreted as a general exam ability indicator. We will next consider adding a second factor to the model. To do this we firstly rerun IGLS to get the same starting values and then add in the factor as follows:

- Change **Estimation method** to **IGLS**.
- Click on the **Start** button.
- Change **Estimation method** to **MCMC**.
- Select **MCMC/Factor Analysis** from the **Model** menu.
- Click on the + button to increase the **Number of factors** to 2.
- Click on the lower + button to **Show Factor** 2.
- Change **Loading 2 value** from 0 to 1.
- Click on the **constraint** boxes for both **Loadings 1** and **2**.
- Click on the **constraint** box for **Variance value** to remove this constraint.

The **Factor Analysis** window should now look as follows:



Number of factors: 2
Show Factor: 2
Factor 2
Random level for factor: 2
Loading 1 value: 0 [✓ Constrained]
Loading 2 value: 1 [✓ Constrained]
Loading 3 value: 0 Constrained
Loading 4 value: 0 Constrained
Loading 5 value: 0 Constrained
Loading 6 value: 0 Constrained
Variance value: 1 Constrained

- Click on the **Set factors** button.
- Click on the **Store Values** button.
- The **Store Factor Chains** window now appears.
- Click on the **Store Factor Loadings** tick box.
- Change column number to **c21**.
- Click on the **Done** buttons on both the **Store Factor Chains** and **Factor Analysis** windows.

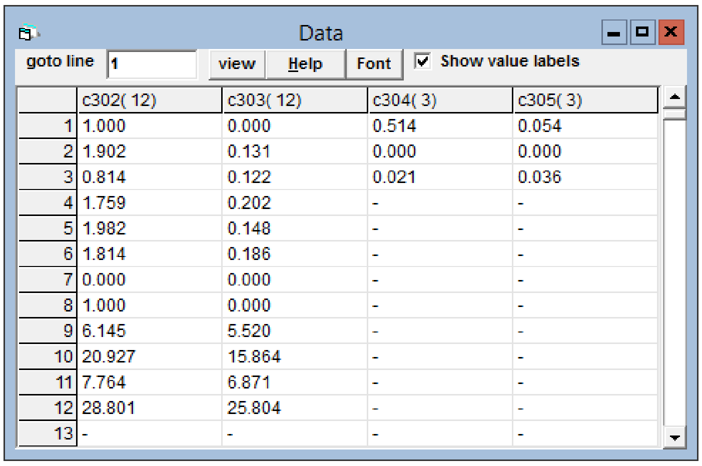
We have now added the second factor to the model and told MLwiN to store the factor loadings stacked in column **c21**. Note that we have not set any correlation between the factors i.e. the correlation is constrained to equal zero. We now need to click on the **Start** button and wait for several minutes for the MCMC method to run.

We can now see in the **Equations** window that again the fixed effect estimates simply capture the means of the 6 responses and are unaffected by the addition of the second factor. We can now output the factors and loadings and look at the estimates (noting that the loadings are available also in the **Equations** window).

- Select **MCMC/Factor Analysis** from the **Model** menu.
- Click on the **Output Options** button.
- The **Factor Analysis Output Options** window will appear.
- On this window click on the three tick boxes to output factor values, loadings and variances respectively.

- Click on the **Calculate** button.
- Click on the **Done** buttons on both the **Factor output** and **Factor** windows.
- Select **View/Edit Data** from the **Data Manipulation** menu.
- Click on the **view** button and select columns **c302**, **c303**, **c304** and **c305** (note you can use the Ctrl key to select multiple columns).
- Click on the **OK** button.

The **Data** window will look as follows:



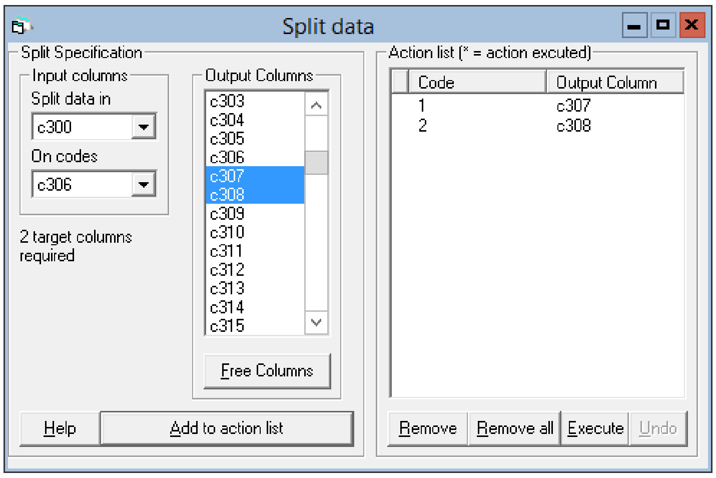
	c302(12)	c303(12)	c304(3)	c305(3)
1	1.000	0.000	0.514	0.054
2	1.902	0.131	0.000	0.000
3	0.814	0.122	0.021	0.036
4	1.759	0.202	-	-
5	1.982	0.148	-	-
6	1.814	0.186	-	-
7	0.000	0.000	-	-
8	1.000	0.000	-	-
9	6.145	5.520	-	-
10	20.927	15.864	-	-
11	7.764	6.871	-	-
12	28.801	25.804	-	-
13	-	-	-	-

Here we have the loadings for the two factors stacked above each other. The first factor still has very similar loadings and can be thought of once again as an indicator of overall test performance. The second factor has some much greater loadings but a much smaller variance (0.021 versus 0.514) and so is explaining a small proportion of the remaining uncertainty. The second factor seems to be picking out students who do comparatively better in the additional tests r4 in biology and r2 in physics. The actual factor values or 'scores' are stored stacked in column **c300** and so we can plot these factors after first splitting the factors into separate columns:

- Select **Generate vector** from the **Data Manipulation** menu.
- Select **Repeated Sequence**.
- Change **Output column** to **c306**.
- Change **Maximum number** to 2.
- Change **Number of repeats per block** to 2439.
- Change **Number of blocks** to 1.
- Click on the **Generate** button.
- Select **Split Column** from the **Data Manipulation** menu.
- Select **Split data in column** to be **c300**.

- Select **On Codes column** to be **c306**.
- Select **Output columns c307** and **c308**.
- Click on the **Add to action list** button.

The screen should look as follows:

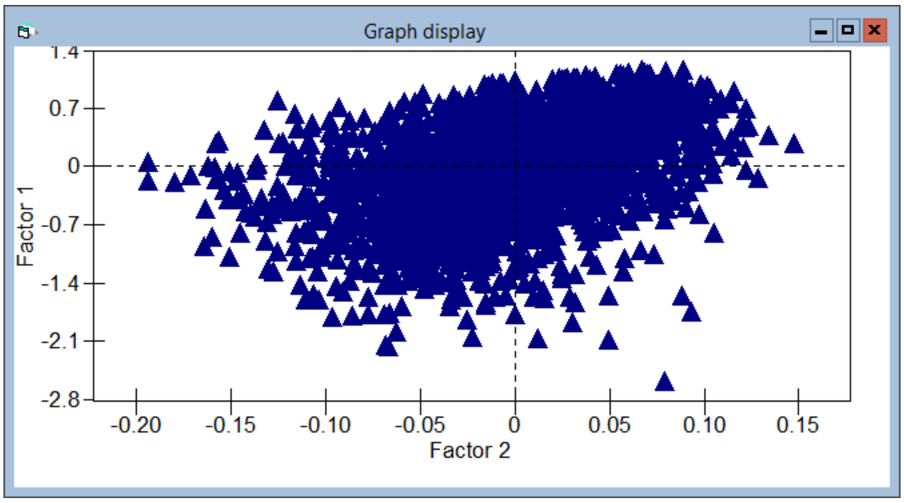


Code	Output Column
1	c307
2	c308

If we now click on the **Execute** button the two sets of factor values will be generated in columns **c307** and **c308**. We can now plot these factors as follows:

- Select **Customised Graph(s)** from the **Graphs** menu.
- Select **c307** as the y variable.
- Select **c308** as the x variable.
- Click on the **Apply** button.

The graph will appear as follows (note that the labels need to be added manually):



Here we see that again pupil 664 is an obvious outlier from the main cloud of points. They have a large positive score for factor 2 due to the fact that they did OK on the physics r2 paper. Another interesting observation is the pupil with the lowest factor 2 value (pupil 1572) who has a factor 1 value of around 0. A close inspection of their scores shows that they did well in the core tests (scores of 10,8 and 8) and OK in the biology r3 test (5) but got zero in the physics r2 test and didn't take the biology r4 test.

20.6 Examining the chains of the loading estimates

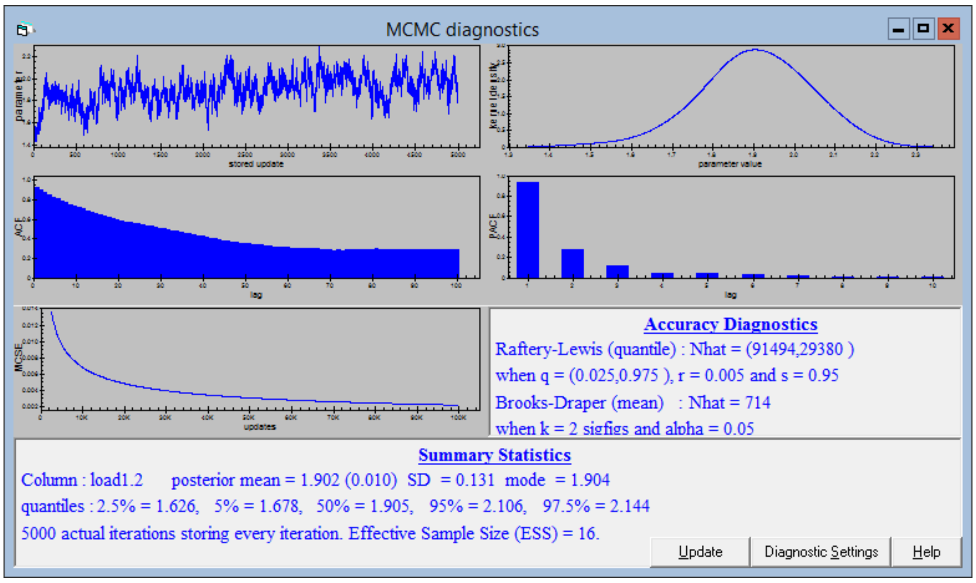
We can see that the second factor is much smaller than the first and is less interpretable. It is important to check convergence of the MCMC chains for all parameters when fitting a factor analysis model to check that our estimates can be trusted. It is also useful to confirm that we have constrained our problem sufficiently to make it identifiable although this is not always obvious from viewing the chains. We have stored the chains of the loadings stacked in column **c21** and so we need to split up the chains and name them. This can be done through the windows in the **Data Manipulation** menu but to save time we will use the **Command Interface** window.

- Select **Command Interface** from the **Data Manipulation** menu. Enter the following commands:

```
▶ CODE 12 1 5000 c29
▶ SPLIT c21 c29 c31-c42
▶ NAME c32 'load1.2'
▶ NAME c39 'load2.3'
```

We can now view the chains for the two named loadings via the **Column Diagnostics** window available from the **Basic Statistics** menu.

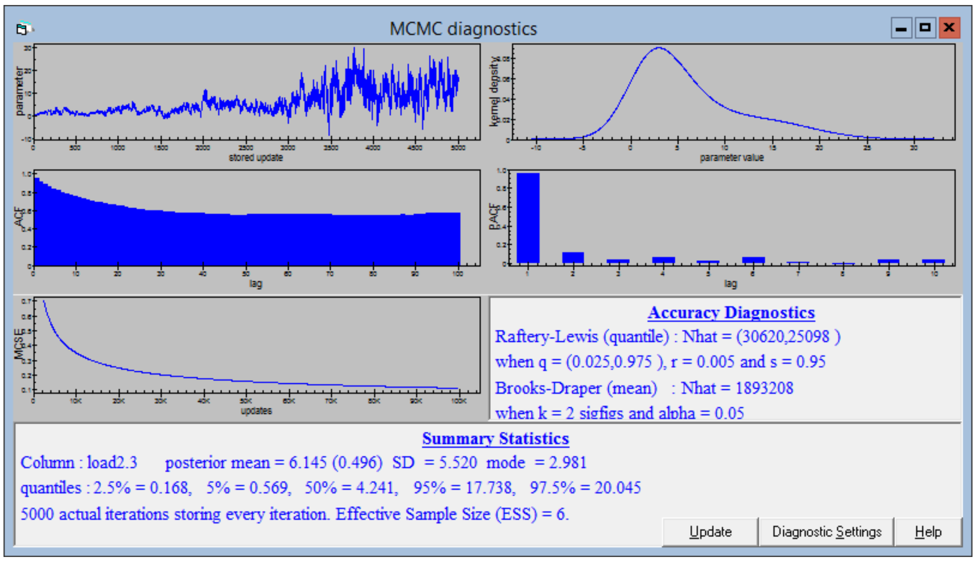
Firstly if we choose column **load1.2** we get the following diagnostics:



Accuracy Diagnostics
Raftery-Lewis (quantile) : Nhat = (91494,29380)
when q = (0.025,0.975), r = 0.005 and s = 0.95
Brooks-Draper (mean) : Nhat = 714
when k = 2 sigfigs and alpha = 0.05

Summary Statistics
Column : load1.2 posterior mean = 1.902 (0.010) SD = 0.131 mode = 1.904
quantiles : 2.5% = 1.626, 5% = 1.678, 50% = 1.905, 95% = 2.106, 97.5% = 2.144
5000 actual iterations storing every iteration. Effective Sample Size (ESS) = 16.

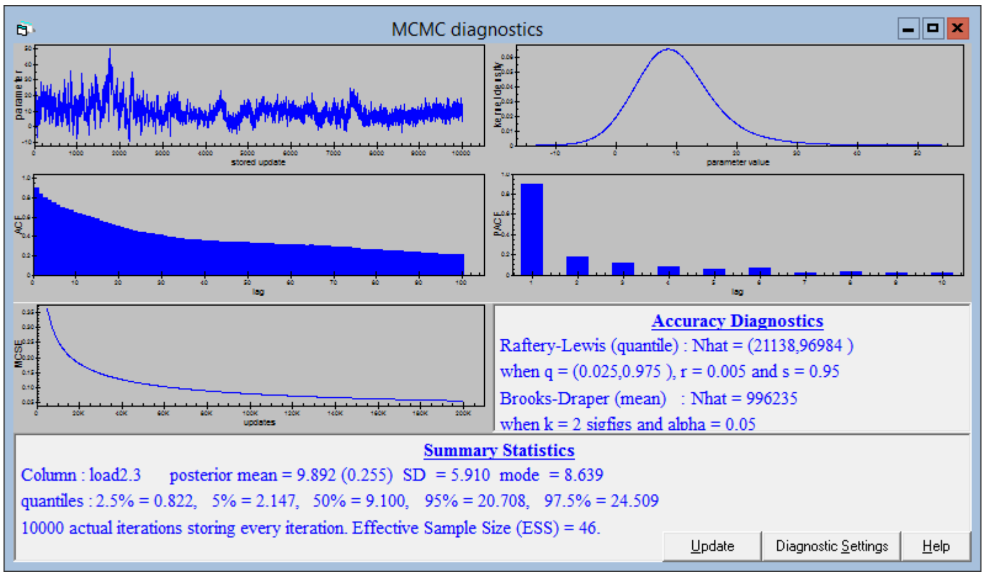
Here we see that we really need to run for longer to get accurate estimates but the chain does at least appear to have converged. If however we look now at the diagnostics for **load2.3** we see the following:



Accuracy Diagnostics
Raftery-Lewis (quantile) : Nhat = (30620,25098)
when q = (0.025,0.975), r = 0.005 and s = 0.95
Brooks-Draper (mean) : Nhat = 1893208
when k = 2 sigfigs and alpha = 0.05

Summary Statistics
Column : load2.3 posterior mean = 6.145 (0.496) SD = 5.520 mode = 2.981
quantiles : 2.5% = 0.168, 5% = 0.569, 50% = 4.241, 95% = 17.738, 97.5% = 20.045
5000 actual iterations storing every iteration. Effective Sample Size (ESS) = 6.

This chain does not seem to have converged yet and the loadings seem to be increasing over time suggesting that perhaps we have not run a long enough burnin for this problem. If we repeat fitting the model (after running IGLS) this time with a burnin of 5000 and a main run of 10,000 and look at the chain (after waiting a long time for the estimation to complete) we see the following:



Accuracy Diagnostics
Raftery-Lewis (quantile) : Nhat = (21138,96984)
when q = (0.025,0.975), r = 0.005 and s = 0.95
Brooks-Draper (mean) : Nhat = 996235
when k = 2 sigfigs and alpha = 0.05

Summary Statistics
Column : load2.3 posterior mean = 9.892 (0.255) SD = 5.910 mode = 8.639
quantiles : 2.5% = 0.822, 5% = 2.147, 50% = 9.100, 95% = 20.708, 97.5% = 24.509
10000 actual iterations storing every iteration. Effective Sample Size (ESS) = 46.

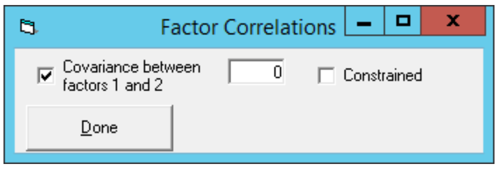
This chain looks a bit better behaved and now due to the longer burnin we have loading estimates that are larger for the second factor thus reducing the importance of the second factor and bringing it into question whether the factor should be included. Note that now the factor variance of the second factor is only about a thousandth of that for the first factor. More research is currently being done to extend the DIC diagnostic to compare factor analysis models.

20.7 Correlated factors

When we fitted the above two factor model we made the assumption that the factors were independent. We can however fit a model where the factors are correlated. To do this in the above model we need to do the following:

- Select **MCMC/Factor Analysis** from the **Model** menu.
- Click on the **Correlate Factors** button.
- On the **Factor Correlations** window tick in the **Covariance between factors 1 and 2** tick box.

The **Factor Correlations** window will look as follows:



As with the other parameters we can constrain this covariance to a particular value. Here we will leave it unconstrained and click on the **Done** button followed by the **Set factors** and **Done** buttons on the **Factor Analysis** window. Given that it appears that the second factor is not needed in this dataset there is probably little point in testing out correlated factors on this model. However for the interested reader if you fit the above model with correlated factors (after running IGLS) and run for 10,000 iterations after a burnin of 5,000 we get an estimate of 0.132 (0.126) for the covariance parameter. The factors appear more balanced with variances of 0.621 and 0.248 respectively. Note here that we have not tested whether the chains have converged or even if the model is identifiable.

An alternative constraint system (that is often used in conventional factor analysis) is to unconstrain the loadings that we have constrained to 1 and instead constrain the factor variances to equal 1. Care has to be taken when using MCMC and such a constraint system, as there is the possibility for the chains to move between the equivalent rotational solutions and hence not converge (see Goldstein & Browne, 2002).

20.8 Multilevel factor analysis

The Hungary Science dataset, as we saw in Chapter 18 has additional structure in that the students are nested within schools. We can clear our current models and set up a multivariate response model with both student and school levels as follows:

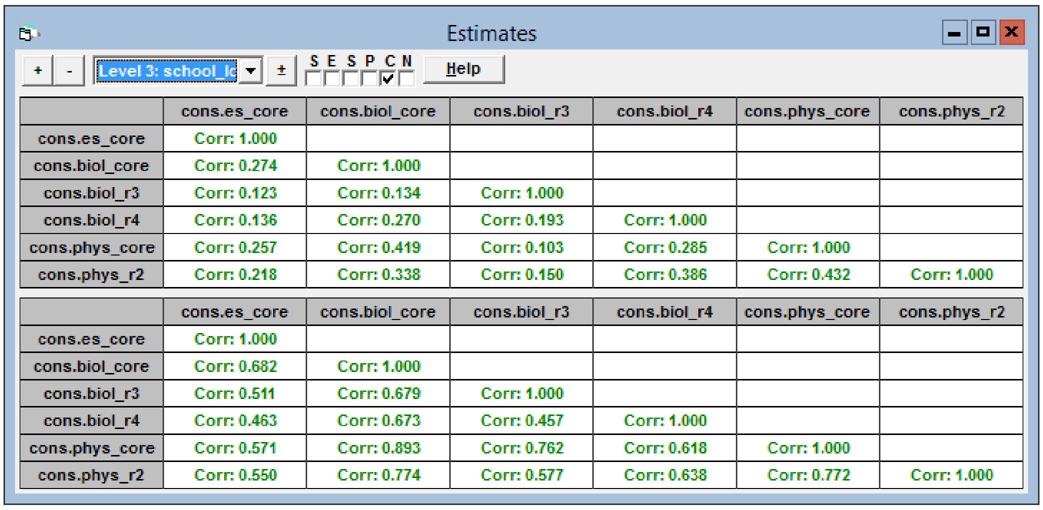
- Change **Estimation method** to **IGLS**.
- Select **Equations** from the **Model** menu and click on the response names.
- On the **Y variable** window change number of levels to **3-ijk**.
- Select **school** for the **level 3(k)** identifier.
- Click on the **Done** button.
- Click on the β_{0j} (**cons_es_core**) and select the **k(school_long)** box.
- Click on the **Done** button.
- Repeat the above for β_{1j}, β_{2j}, β_{3j}, β_{4j} and β_{5j}.
- Click on Ω_u and select **set full matrix** to put back the covariances.
- Click on the **Start** button to run the model.

After the IGLS model converges we can look at the correlation matrices at both levels via the **Estimate Tables** window:

- Select **Estimate Tables** from the **Model** menu.

- Replace **Fixed Part** with **Level 2: student_long** in the pull down list.
- Remove the tick under S, E, S and P by clicking in the boxes and add a tick under the C.
- Click on the + button and replace **Level 2: student_long** with **Level 3: school_long**.

The correlation matrices will now be displayed (individual level on top) as follows:



	cons.es_core	cons.biol_core	cons.biol_r3	cons.biol_r4	cons.phys_core	cons.phys_r2
cons.es_core	Corr: 1.000					
cons.biol_core	Corr: 0.274	Corr: 1.000				
cons.biol_r3	Corr: 0.123	Corr: 0.134	Corr: 1.000			
cons.biol_r4	Corr: 0.136	Corr: 0.270	Corr: 0.193	Corr: 1.000		
cons.phys_core	Corr: 0.257	Corr: 0.419	Corr: 0.103	Corr: 0.285	Corr: 1.000	
cons.phys_r2	Corr: 0.218	Corr: 0.338	Corr: 0.150	Corr: 0.386	Corr: 0.432	Corr: 1.000

	cons.es_core	cons.biol_core	cons.biol_r3	cons.biol_r4	cons.phys_core	cons.phys_r2
cons.es_core	Corr: 1.000					
cons.biol_core	Corr: 0.682	Corr: 1.000				
cons.biol_r3	Corr: 0.511	Corr: 0.679	Corr: 1.000			
cons.biol_r4	Corr: 0.463	Corr: 0.673	Corr: 0.457	Corr: 1.000		
cons.phys_core	Corr: 0.571	Corr: 0.893	Corr: 0.762	Corr: 0.618	Corr: 1.000	
cons.phys_r2	Corr: 0.550	Corr: 0.774	Corr: 0.577	Corr: 0.638	Corr: 0.772	Corr: 1.000

Here we see that the correlations at the individual level now range from 0.123 to 0.432 and have generally been reduced slightly by the introduction of school effects. At the school level the correlations are greater ranging from 0.457 to 0.893. We will now try and explain these correlations with two factors, one at each level.

20.9 Two level factor model

To fit the two level factor model we firstly need to remove all the covariances from the model:

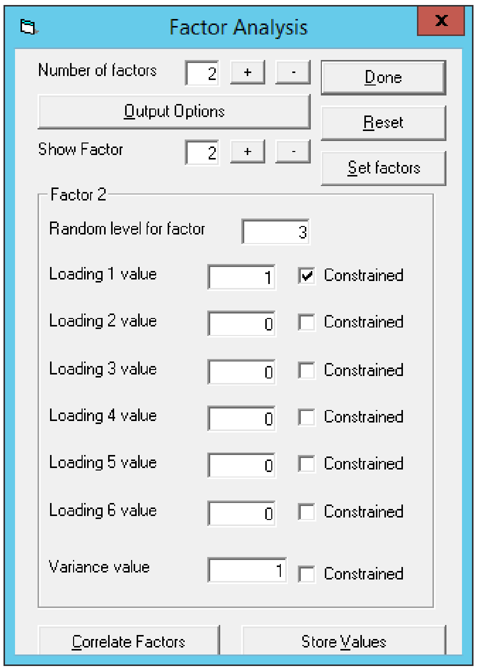
- Select **Equations** from the **Model** menu.
- In the **Equations** window click on the **Estimates** button until the estimates appear as green numbers.
- Scroll down until Ω_v is visible.
- Click on the Ω_v and select **set diagonal matrix**.
- Scroll down until Ω_u is visible.

- Click on the Ω_u and select **set diagonal matrix**.
- When finished click on the **Start** button.

We have effectively run (using IGLS) 6 separate variance components models, one for each response. We now need to change to MCMC and add the factors.

- Change **Estimation method** to **MCMC**.
- Select **MCMC/Factor Analysis** from the **Model** menu.
- Click on the **Reset** button to remove the existing factors.
- Click on the + button twice to set **Number of factors** to 2.
- Set up the first factor as before i.e. have the first loading constrained to equal 1 and no other constraints.
- Click on the **Show Factor** + button to view factor 2.
- Change **Random level for factor** to 3 (school level)
- Change **Loading 1 value** to 1 and constrain it while removing any other constraints

The **Factor Analysis** window for the second factor will look as shown below and we now need to click on the **Set factors** and **Done** buttons to finish.

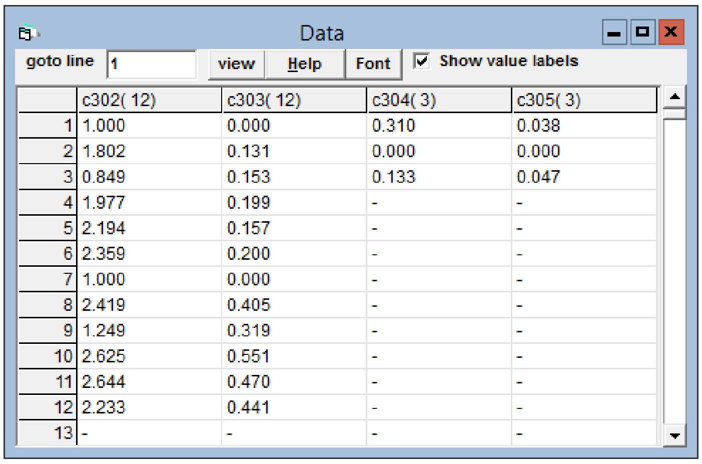


Number of factors 2
Show Factor 2
Factor 2
Random level for factor 3
Loading 1 value 1 [✓ Constrained]
Loading 2 value 0 Constrained
Loading 3 value 0 Constrained
Loading 4 value 0 Constrained
Loading 5 value 0 Constrained
Loading 6 value 0 Constrained
Variance value 1 Constrained

We now need to run the model using MCMC which we will run for the default settings of a burn-in of 500 iterations and main run of 5,000 iterations so you may have to reset these values on the **Estimation Control** window. Click on the **Start** button to run the model. After a few minutes the estimation will complete and we can output the estimates for the factors:

- Select **MCMC/Factor Analysis** from the **Model** menu.
- Click on the **Output Options** button.
- The **Factor Analysis Output Options** window will appear.
- On this window click on the three tick boxes to output factor values, loadings and variances respectively.
- Click on the **Calculate** button.
- Click on the **Done** buttons on both the **Factor Output** and **Factor** windows.
- Select **View/Edit Data** from the **Data Manipulation** menu.
- Click on the **view** button and select columns **c302**, **c303**, **c304** and **c305**.
- Note you can use the Ctrl key to select multiple columns.
- Click on the **OK** button.

The **Data** window will appear as follows:



	c302(12)	c303(12)	c304(3)	c305(3)
1	1.000	0.000	0.310	0.038
2	1.802	0.131	0.000	0.000
3	0.849	0.153	0.133	0.047
4	1.977	0.199	-	-
5	2.194	0.157	-	-
6	2.359	0.200	-	-
7	1.000	0.000	-	-
8	2.419	0.405	-	-
9	1.249	0.319	-	-
10	2.625	0.551	-	-
11	2.644	0.470	-	-
12	2.233	0.441	-	-
13	-	-	-	-

Here the first 6 elements in column **c302** are the loadings for the factor at the student level. These are very similar to the first factor model. The second six numbers are the loadings for the factor at the school level. Here we see again that all six loadings are positive and so we can think of these as producing a factor that represents overall exam achievement at the school level. The variance of this factor is half the size of the variance for the factor at the student level.

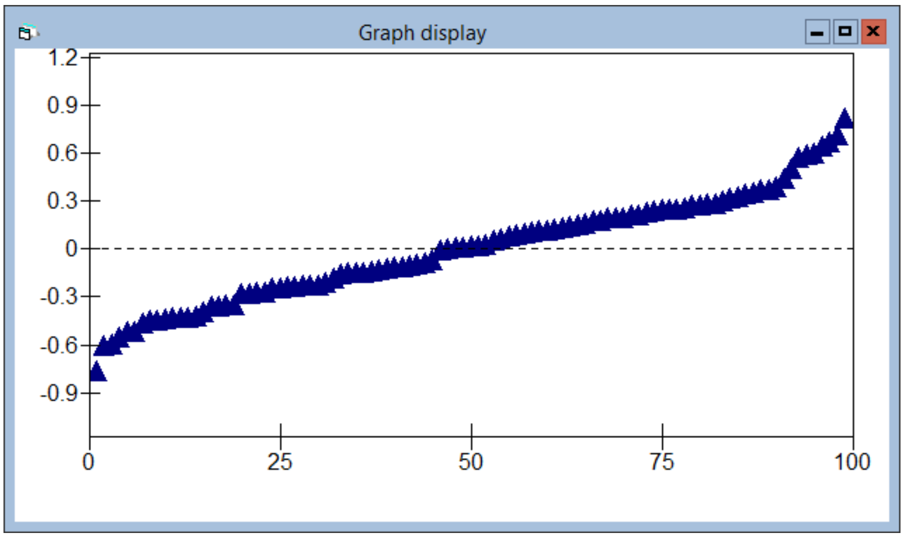
The factor scores themselves are stacked in **c300** by default and to avoid confusion are enlarged to be the length of the observation (student) level (2439) i.e. there are 2439 scores for the level 1 factor, 99 for the level 2 factor followed by 2340 zeroes. To capture the factors for the 99 schools we therefore need to get hold of the 2440th to 2538th elements of **c300** and then rank them. We can do this and plot them as follows:

- Select **Command interface** from the **Data Manipulation** menu. Enter the following command:

```
▶ PICK 2440 2538 c300 c30
▶ RANK c30 c31
```

- Select **Customised Graph(s)** from the **Graphs** menu.
- Select **c30** as the y variable.
- Select **c31** as the x variable.
- Click on the **Apply** button.

The graph of the 99 school factor values is then as follows:



Here school number 8033 has the highest value of the factor while school number 2089 has the lowest and so the interested reader can look at the marks for the children in these schools to confirm that this can be used as an indicator of school achievement.

20.10 Extensions and some warnings

We could of course now extend the last model to include a second factor at the school level and interpret this factor. We also have not considered fitting more terms in the fixed part of the model. We could consider fitting the gender variable as a fixed effect for some or all of our responses. There are some restrictions in the models that can be fitted in MLwiN for example the variance matrix at the level which factors are included must be diagonal and have only one random term per response. The software will (hopefully) check if a model is valid.

We will end this chapter by reiterating a warning. Unlike many of the multilevel models that we discussed in earlier chapters multilevel factor analysis models should be used with caution. The user should take care that firstly their model is identifiable and secondly that their interpretation is unique and/or appropriate. As these models are an area that we have not researched deeply there is a certain amount of using these models at your own risk and we will emphasize the importance of looking at the parameter chains to confirm convergence in these models.

Chapter learning outcomes

- ⋆ What is meant by a factor analysis model.
- ⋆ How to fit simple Bayesian factor analysis models in MLwiN.
- ⋆ How to interpret factor loadings and values.
- ⋆ How to fit more than one factor.
- ⋆ How to fit multilevel factor analysis models.

Chapter 21

Using Structured MCMC

In this chapter we consider alternative MCMC methods for fitting Normal response multilevel models that will aim to reduce the correlations in the chains produced for the fixed effects and residuals. One reason for 'poor mixing' of chains is correlation between parameters that are updated in different blocks in the standard Gibbs sampling algorithm. Structured MCMC methods circumvent this problem by updating certain groups of parameters together in bigger blocks. The methods that have been implemented in the new version of MLwiN are limited to Normal response models and in fact to the subset of 2-level nested Normal response models with no complex variation at level 1. Thus one should consider these methods as illustrative of alternative methodology rather than of huge benefit to the user base. We will consider three models in this chapter, firstly a variance components model with no predictors, second a random intercepts model and finally a random slopes model. We will cover slightly more of the technical details than in earlier chapters so some readers might prefer to skip some of the detailed mathematics in the next section and move on to the later practical sections.

21.1 SMCMC Theory

We will in this chapter consider the **tutorial** example considered earlier where we have exam marks of pupils clustered as they are taught in groups in a set of schools.

Let y_{ij} be the mark for pupil i in school j, then perhaps the simplest model we could fit is

$$y_{ij} = \beta_0 + u_j + e_{ij}, \quad u_j \sim \mathrm{N}(0, \sigma_u^2), \quad e_{ij} \sim \mathrm{N}(0, \sigma_e^2)$$

Here we have an estimated average mark β_0, school level residuals u_j with variance σ_u^2 and pupil level residuals e_{ij} with variance σ_e^2. We will assume that we have J schools and that school j has n_j pupils with the total number of pupils being N.

For a Bayesian model we will use 'diffuse' priors as follows:

$$p(\beta_0) \propto 1, \quad p(1/\sigma_u^2) \sim \Gamma(\varepsilon, \varepsilon), \quad p(1/\sigma_e^2) \sim \Gamma(\varepsilon, \varepsilon)$$

In this example with conjugate priors we have full conditionals that can be easily evaluated and hence used in a Gibbs sampling algorithm as follows:

Step 1: $\beta_0 \sim \mathrm{N}(\hat{\beta}_0, \hat{D}_0)$ where $\hat{D}_0 = \frac{\sigma_e^2}{N}$ and $\hat{\beta}_0 = \frac{\sum_{i,j}(y_{ij}-u_j)}{N}$

Step 2: $u_j \sim \mathrm{N}(\hat{u}_j, \hat{D}_j)$ where $\hat{D}_j = (\frac{n_j}{\sigma_e^2} + \frac{1}{\sigma_u^2})^{-1}$ and $\hat{u}_j = \frac{\hat{D}_j}{\sigma_e^2} \sum_{i=1}^{n_j} (y_{ij} - \beta_0)$

Step 3: $1/\sigma_u^2 \sim \Gamma(c_u, d_u)$ gives $c_u = J/2 + \varepsilon$ and $d_u = \sum_{j=1}^{J} \frac{u_j^2}{2} + \varepsilon$

Step 4: $1/\sigma_e^2 \sim \Gamma(c_e, d_e)$ gives $c_e = N/2 + \varepsilon$ and $d_e = \sum_{i,j} \frac{e_{ij}^2}{2} + \varepsilon$, where $e_{ij} = y_{ij} - \beta_0 - u_j$

One source of 'poor mixing' occurs due to posterior correlation between the residuals u_j and the intercept β_0. Basically the intercept for any specific school j is the sum of the overall intercept β_0 and the school residual u_j and so if the value of the overall intercept increases then the school residuals tend to decrease to still fit the data and so there is a negative correlation between the two parameters.

To counter this problem the technique of structured MCMC sampling was developed by Sargent et al. (2000). They describe SMCMC as a 'general method for Bayesian computing in richly-parameterized models, based on a blocked hybrid of the Gibbs sampling and Metropolis Hastings algorithms'. Basically they formulated an algorithm that treats all fixed effects and residuals as one large vector that will be updated in one step as it has a multivariate Normal conditional posterior distribution as detailed below:

We write the model as a single vector response multivariate Normal model but the y vector is augmented by rearranging the terms for the level 2 effects to give

$$y^* \sim \mathrm{MVN}(X^* \boldsymbol{\beta}^*, V^*)$$

where we effectively rearrange $\mathbf{u}_j \sim \mathrm{MVN}(0, \Omega_u)$ as $0 \sim \mathrm{MVN}(-\mathbf{u}_j, \Omega_u)$ and then add these zeroes to create y^*.

Note that $\boldsymbol{\beta}^*$ is a vector containing β_0 and all elements of $\mathbf{u}_j$.

This model is just a rearrangement of the multilevel model and so has the constraints built in.

The basic premise for Normal response models is to be able to write the model in multivariate Normal form:

$$Y = X\boldsymbol{\theta} + E, \quad \mathrm{cov}(E) = \Lambda$$

Then

$$p(\boldsymbol{\theta}|Y, X, \Lambda) \sim \mathrm{MVN}\left((X^T\Lambda^{-1}X)^{-1}X^T\Lambda^{-1}Y, (X^T\Lambda^{-1}X)^{-1}\right)$$

The method updates all fixed and random effects in one step and hence improves mixing of the MCMC algorithm but relies on 'nice' forms for the large matrices that need inverting.

We will now return to the **tutorial** example that consists of data on 4059 (N) students in 65 (J) schools, the response of interest is (Normalised) exam score at 16.

We start by reparameterising the model into a centred formulation (as will be described in more detail in later chapters) by letting $u_j^* = \beta_0 + u_j$ and so we have:

$$y_{ij} = u_j^* + e_{ij}, \quad u_j^* \sim \mathrm{N}(\beta_0, \sigma_u^2), \quad e_{ij} \sim \mathrm{N}(0, \sigma_e^2)$$

Then reconstructing the random effects statement as $0 \sim N(\beta_0 - u_j^*, \sigma_u^2)$ we can construct our model in the form

$$Y = X\boldsymbol{\theta} + E, \quad \mathrm{cov}(E) = \Lambda$$

Here we have

$$Y = \left(\frac{y}{0_J}\right) = \left[\begin{array}{cccc|c} 1_{n_1} & 0_{n_1} & \dots & 0_{n_1} & \\ 0_{n_2} & 1_{n_2} & \dots & 0_{n_2} & \\ \vdots & \vdots & \ddots & \vdots & 0_N \\ 0_{n_J} & 0_{n_J} & \dots & 1_{n_J} & \\ \hline & -I_J & & 1_J & \end{array}\right] \left(\begin{array}{c} u_1^* \\ \vdots \\ u_J^* \\ \beta_0 \end{array}\right) + \left(\frac{e}{\delta}\right)$$

with

$$\Lambda = \left(\begin{array}{ll} \sigma_e^2 I_N & 0 \\ 0 & \sigma_u^2 I_J \end{array}\right)$$

Here we define 0_m and 1_n to be column vectors of m 0s and n 1s respectively.

Recall we now have:

$$p(\boldsymbol{\theta}|Y,X,\Lambda) \sim \text{MVN}\left((X^T\Lambda^{-1}X)^{-1}X^T\Lambda^{-1}Y, (X^T\Lambda^{-1}X)^{-1}\right)$$

where

$$X^T\Lambda^{-1}X = \left[\begin{array}{ccc|c} 1/\sigma_u^2 + n_1/\sigma_e^2 & 0 & 0 & -1/\sigma_u^2 \\ 0 & \ddots & 0 & \vdots \\ 0 & 0 & 1/\sigma_u^2 + n_J/\sigma_e^2 & -1/\sigma_u^2 \\ \hline -1/\sigma_u^2 & \cdots & -1/\sigma_u^2 & J/\sigma_u^2 \end{array}\right]$$

Note that we could simply invert this 66×66 matrix by standard matrix inversion, however we can use the form of the matrix and some results from matrix algebra to speed up the inversion. The following matrix inversion result can be used when inverting $X^T\Lambda^{-1}X$:

$$\left[\begin{array}{cc} A & B \\ B^T & C \end{array}\right]^{-1} = \left[\begin{array}{cc} A^{-1} & 0 \\ 0 & 0 \end{array}\right] +$$

$$\left[\begin{array}{c} -A^{-1}B \\ I \end{array}\right]\left[\begin{array}{c} C - B^TA^{-1}B \end{array}\right]^{-1}\left[\begin{array}{cc} -B^TA^{-1} & I \end{array}\right]$$

In our example the submatrix A is diagonal and so is very easy to invert and this reduces the algorithm time.

The update steps for σ_u^2 and σ_e^2 are as in the standard Gibbs sampling algorithm.

21.2 Fitting the model using MLwiN

We will first set up the above model in MLwiN and fit it using the standard MCMC methods used in previous chapters. Firstly we need to retrieve the worksheet:

- Select **Open sample worksheet** from the **File** menu.
- Select **tutorial.ws**.

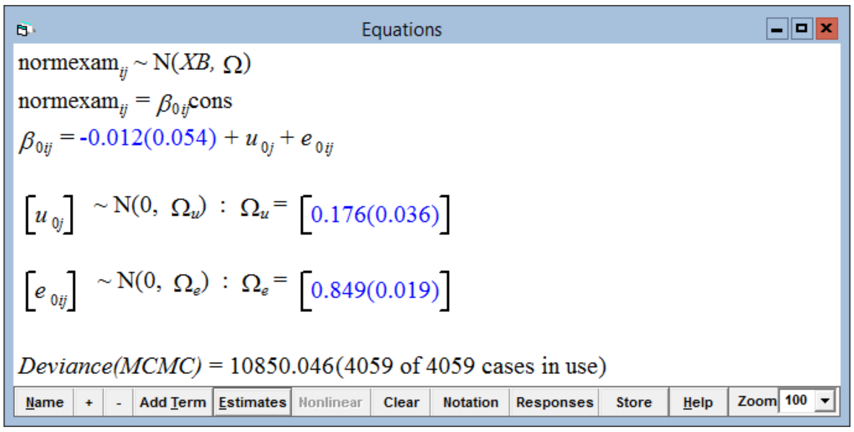
This will open the **Names** window and we now need to bring up the **Equations** window and set up the model.

- Select **Equations** from the **Model** menu.
- Click on **y** (either of the **y** symbols shown will do).
- In the **y** list, select **normexam**.
- In the **N levels** list, select **2-ij**.
- In the **level 2(j):** list, select **school**.
- In the **level 1(i):** list, select **student**.
- Click on the **Done** button.
- Click on the red x_0.
- In the drop-down list, select **cons**.
- Check the box labelled **i(student)**.
- Check the box labelled **j(school)**.
- Click on the **Done** button.

These commands will set up the model which we now need to fit using MCMC, although firstly as usual we will fit it using IGLS to get starting values.

- Click **Start**.
- Click on the **Estimation Control** button
- Select the tab labelled **MCMC**
- Click **Start**.

Upon running the model the **Equations** window should look as follows:



$\text{normexam}_{ij} \sim \text{N}(XB, \Omega)$

$\text{normexam}_{ij} = \beta_{0ij}\text{cons}$

$\beta_{0ij} = -0.012(0.054) + u_{0j} + e_{0ij}$

$\left[u_{0j}\right] \sim \text{N}(0, \Omega_u) : \Omega_u = \left[0.176(0.036)\right]$

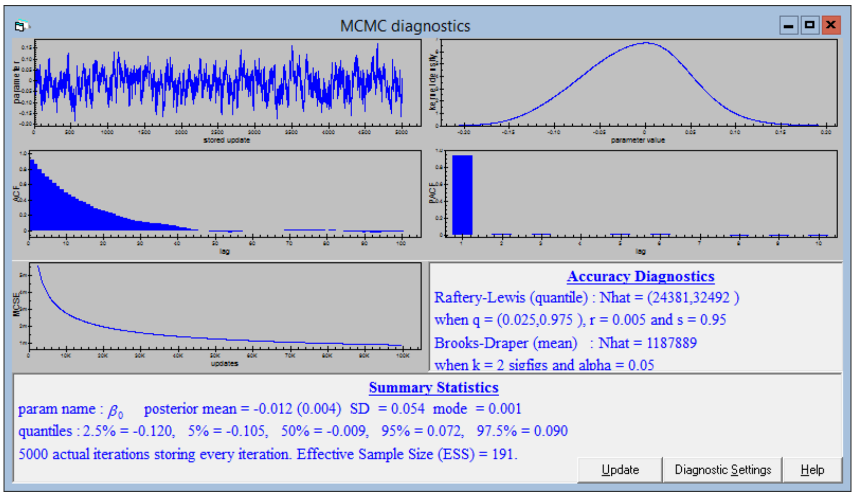
$\left[e_{0ij}\right] \sim \text{N}(0, \Omega_e) : \Omega_e = \left[0.849(0.019)\right]$

Deviance(MCMC) = 10850.046(4059 of 4059 cases in use)

We are interested in this chapter in looking at how well the chains produced by the MCMC methods mix and so for this we would like to look at the output for the intercept parameter β_0. For this we need to get diagnostics via the **Trajectories** window.

- Select **Trajectories** from the **Model** menu.
- Click on the chain for β_0 in the **Trajectories** window.
- On the window that appears click on the **Yes** button.

The **MCMC diagnostics** window will appear as follows:



Accuracy Diagnostics
Raftery-Lewis (quantile) : Nhat = (24381,32492)
when q = (0.025,0.975), r = 0.005 and s = 0.95
Brooks-Draper (mean) : Nhat = 1187889
when k = 2 sigfigs and alpha = 0.05

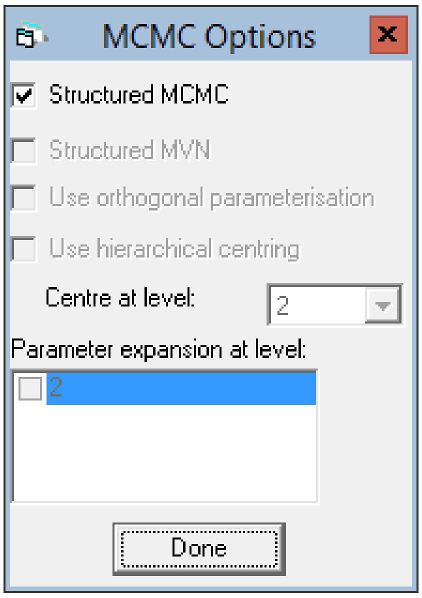
Summary Statistics
param name : β_0 posterior mean = -0.012 (0.004) SD = 0.054 mode = 0.001
quantiles : 2.5% = -0.120, 5% = -0.105, 50% = -0.009, 95% = 0.072, 97.5% = 0.090
5000 actual iterations storing every iteration. Effective Sample Size (ESS) = 191.

Looking at this window we see that the mixing of the chain is poor with an ACF plot showing high autocorrelations and an effective sample size of only 191.

We will next compare the results produced when using the SMCMC code that has been added to MLwiN. For this we will need to rerun IGLS and choose **SMCMC** from the new **MCMC Options** window

- Click on the **IGLS/RIGLS** tab on the **Estimation control** window.
- Click **Start**.
- Click on the **Estimation Control** button.
- Select the tab labelled **MCMC** on the **Estimation control** window.
- Select **MCMC/MCMC options** from the **Model** menu.
- Click on **Structured MCMC**.

The **MCMC options** window should then look as follows:



[✓] Structured MCMC
[ ] Structured MVN
[ ] Use orthogonal parameterisation
[ ] Use hierarchical centring
Centre at level: 2
Parameter expansion at level: [ ] 2

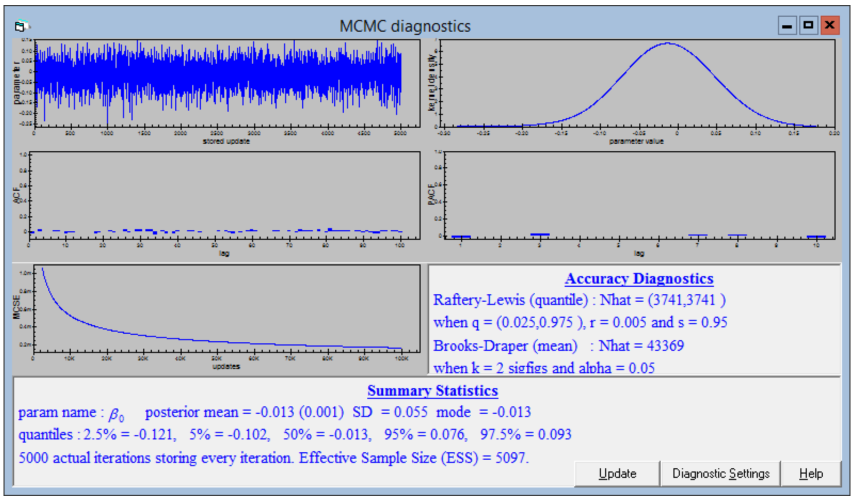
We will now run the model using SMCMC by doing the following:

- Click on the **Done** button on the **MCMC options** window.
- Click **Start**.

After the estimation finishes we need to view the chain for β_0.

- Click on the chain for β_0 in the **Trajectories** window.
- On the window that appears click on the **Yes** button.

We then get the following window:



Accuracy Diagnostics
Raftery-Lewis (quantile) : Nhat = (3741,3741)
when q = (0.025,0.975), r = 0.005 and s = 0.95
Brooks-Draper (mean) : Nhat = 43369
when k = 2 sigfigs and alpha = 0.05

Summary Statistics
param name : β_0 posterior mean = -0.013 (0.001) SD = 0.055 mode = -0.013
quantiles : 2.5% = -0.121, 5% = -0.102, 50% = -0.013, 95% = 0.076, 97.5% = 0.093
5000 actual iterations storing every iteration. Effective Sample Size (ESS) = 5097.

Here we see a much better mixing chain, in fact we are getting effectively independent samples for the parameter and an effective sample size (ESS) of slightly more than our actual sample size! This is due to a slight negative autocorrelation in the chain. It is clear therefore with this example that SMCMC has a great benefit in terms of chain mixing, however this has to be balanced with a computational cost in terms of computation time.

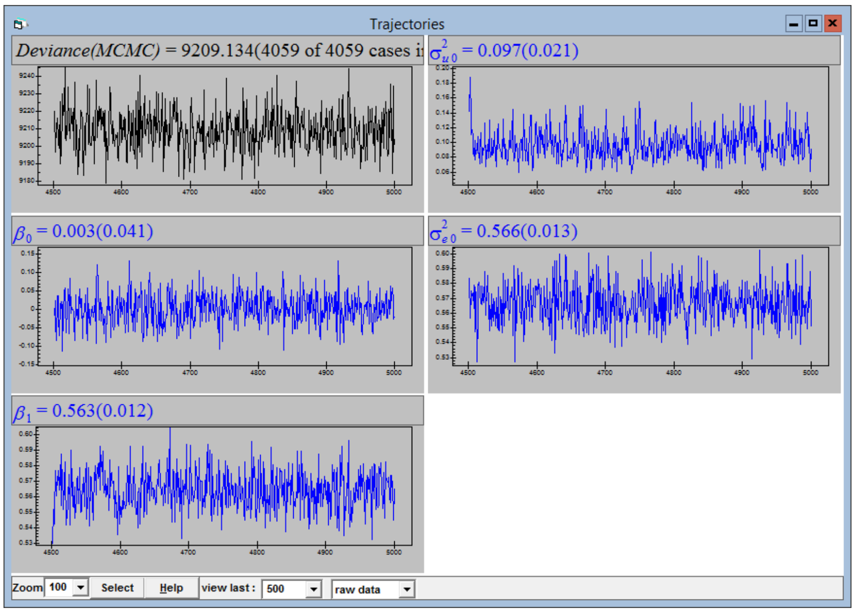
With all windows closed the standard algorithm takes just over 3 seconds to run this model, whilst the SMCMC method takes 4.5 seconds. For this simple model the computational overhead is small but as we see this will increase with later models.

21.3 A random intercepts model

In Chapter 3 of this manual we first looked at a random intercepts model that also contained the predictor **standlrt**. We can compare how SMCMC does when we add in this predictor to the model. Firstly we need to set up the model:

- Click on the **IGLS/RIGLS** tab on the **Estimation control** window.
- Click the **Add Term** button on the **Equations** window tool bar.
- Select **standlrt** from the **variable** list.
- Click on the **Done** button.
- Click **Start**.
- Select the tab labelled **MCMC** on the **Estimation control** window.
- Click **Start**.

After the model runs the **Trajectories** window will look as follows:


Trajectories
Deviance(MCMC) = 9209.134(4059 of 4059 cases i
σ^2_{u0} = 0.097(0.021)
β_0 = 0.003(0.041)
σ^2_{e0} = 0.566(0.013)
β_1 = 0.563(0.012)
Zoom 100
Select
Help
view last : 500
raw data

If we compare this window with the equivalent window in chapter 3 we see the chains are mixing better, in particular the β_0 chain. In fact if we investigate the ESS for the parameters we see that for β_0 this rises from 216 to 5209 and for β_1 from 4413 to 5332. So once again we observe essentially independent sampling for the parameters with a slight computational cost of a run time of 5 seconds as opposed to 3.5 seconds.

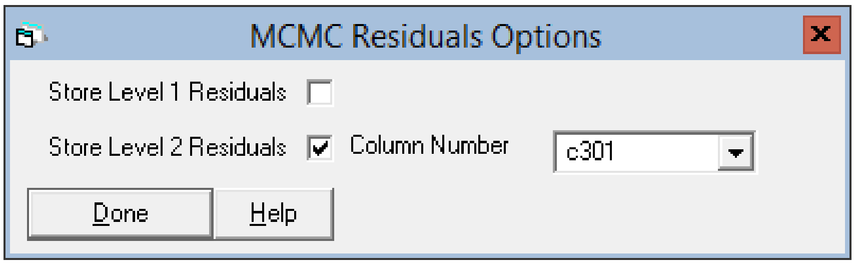
21.4 Examining the residual chains

The SMCMC method is updating the level 2 residuals as well as the fixed effects in the same step and so there should also be a benefit to the mixing of these chains. The residual chains are not normally stored by default by MLwiN but a method to store them is shown in section 4.6.

We will here make use of the **Store residuals** window as described in chapter 4.

- Click on the **IGLS/RIGLS** tab on the **Estimation control** window.
- Click **Start**.
- Click on the **MCMC** tab on the **Estimation control** window.
- Select **MCMC/Store Residuals** from the **Model** menu.
- Click on the **Store Level 2 residuals** tickbox.

The window should now look as follows:

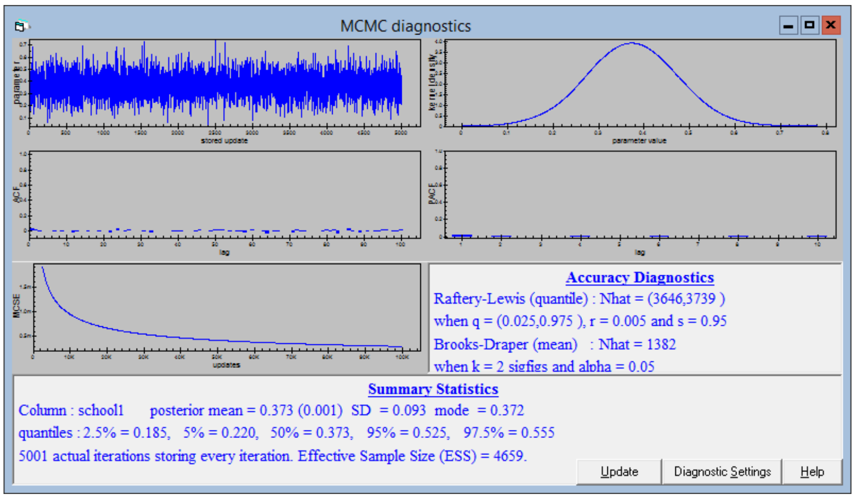

MCMC Residuals Options

Store Level 1 Residuals ☐

Store Level 2 Residuals ☑ Column Number c301

- Click on the **Done** button on the **Store Residuals** window.
- Select the tab labelled **MCMC** on the **Estimation control** window.
- Change the length of **monitoring chain** to 5001.
- Click **Start**.

The estimation procedure will now store residuals as it proceeds through the iterations using SMCMC. Chapter 4 gives instructions on how to extract individual residual chains. For brevity we do not repeat the instructions here. If however you follow the instructions and then view the chain for school 1's residual you will see the following:



Accuracy Diagnostics
Raftery-Lewis (quantile) : Nhat = (3646,3739)
when q = (0.025,0.975), r = 0.005 and s = 0.95
Brooks-Draper (mean) : Nhat = 1382
when k = 2 sigfigs and alpha = 0.05

Summary Statistics
Column : school1 posterior mean = 0.373 (0.001) SD = 0.093 mode = 0.372
quantiles : 2.5% = 0.185, 5% = 0.220, 50% = 0.373, 95% = 0.525, 97.5% = 0.555
5001 actual iterations storing every iteration. Effective Sample Size (ESS) = 4659.

Here we see that the ESS for this residual is 4659 as opposed to 1879 from the standard method showing once again the benefit of Structured MCMC.

21.5 Random slopes model theory

Adding additional predictors as fixed effects to a model, as we did in the last model, only increases the size of the multivariate vector used in SMCMC by 1 per predictor. Allowing that predictor to be random at the school level and thus have different effects for each school will add lots of predictors to the vector, 65 in the tutorial example.

In the last section we introduced one predictor, **standlrt**, a reading test at age 11 (x) and here we allow its coefficient to vary across schools as follows:

$$y_{ij} = \beta_0 + \beta_1 x_{ij} + u_{0j} + u_{1j} x_{ij} + e_{ij}$$
$$\mathbf{u}_j \sim \text{MVN}(0, \Omega_u), \quad e_{ij} \sim \text{N}(0, \sigma_e^2)$$

where $\mathbf{u}_j = (u_{0j}, u_{1j})^T$ and Ω_u is the variance/covariance matrix at the school level which consists of the following terms: $\text{var}(u_{0j}) = \sigma_{u0}^2$, $\text{var}(u_{1j}) = \sigma_{u1}^2$ and $\text{cov}(u_{0j}, u_{1j}) = \sigma_{u01}$.

For a Bayesian model we will use 'diffuse' priors as follows:

$$p(\beta_0) \propto 1, \quad p(\beta_1) \propto 1, \quad p(\Omega_u) \sim \text{IWishart}(\nu_p, S_p), \quad p(1/\sigma_e^2) \sim \Gamma(\varepsilon, \varepsilon)$$

In this example with conjugate priors we have full conditionals that can be easily evaluated and hence used in a Gibbs sampling algorithm as follows:

Step 1: $\boldsymbol{\beta} \sim \text{MVN}(\hat{\boldsymbol{\beta}}, \hat{\Sigma}_\beta)$ gives $\hat{\Sigma}_\beta = \sigma_e^2(X^TX)^{-1}$ and $\hat{\boldsymbol{\beta}} = (X^TX)^{-1}X^Td$ where $d_{ij} = y_{ij} - \mathbf{X}_{ij}^T\mathbf{u}_j$.

Step 2: $\mathbf{u}_j \sim \text{MVN}(\hat{\mathbf{u}}_j, \hat{\Sigma}_j)$ where $\hat{\Sigma}_j = \left[\sum\limits_{i=1}^{n_j} \frac{\mathbf{X}_{ij}^T\mathbf{X}_{ij}}{\sigma_e^2} + \Omega_u^{-1}\right]^{-1}$ and $\hat{\mathbf{u}}_j = \frac{\hat{\Sigma}_j}{\sigma_e^2}\sum\limits_{i=1}^{n_j} \mathbf{X}_{ij}^T(y_{ij} - \mathbf{X}_{ij}^T\boldsymbol{\beta})$.

Step 3: $\Omega_u^{-1} \sim \text{Wishart}_2(\nu_u, S_u)$ gives $\nu_u = J + \nu_p$ and $S_u = (\sum\limits_{j=1}^{J} \mathbf{u}_j\mathbf{u}_j^T + S_p^{-1})^{-1}$.

Step 4: $1/\sigma_e^2 \sim \Gamma(c_e, d_e)$ gives $c_e = N/2 + a_e$ and $d_e = \sum_{i,j} \frac{e_{ij}^2}{2} + b_e$ where $e_{ij} = y_{ij} - \mathbf{X}_{ij}^T\boldsymbol{\beta} - \mathbf{X}_{ij}^T\mathbf{u}_j$.

This is the standard algorithm used in MLwiN for this model. For SMCMC steps 3 and 4 are as for the standard algorithm and we will have one step to replace steps 1 and 2 with a 132 dimensional Normal update.

We start by reparameterising the model into a centred formulation by letting $u_{0j}^* = \beta_0 + u_{0j}$ and $u_{1j}^* = \beta_1 + u_{1j}$ so we have:

$$y_{ij} = u_{0j}^* + x_{ij}u_{1j}^* + e_{ij}, \quad \mathbf{u}_j^* \sim \text{MVN}(\boldsymbol{\beta}, \Omega_u), \quad e_{ij} \sim \text{N}(0, \sigma_e^2)$$

Then reconstructing the random effects statements as previously we can construct our model in the form

$$Y = X\boldsymbol{\theta} + E, \quad \text{cov}(E) = \Lambda$$

Here we have

$$Y = \left(\frac{y}{0_{2J}}\right) = \left[\begin{array}{ccccc|c} 1_{n_1} & x_{.1} & 0 & 0 & 0 & \\ 0 & 0 & \ddots & 0 & 0 & 0_N \quad 0_N \\ 0 & 0 & 0 & 1_{n_J} & x_{.J} & \\ \hline \multicolumn{2}{c}{-I_2} & 0 & 0 & 0 & I_2 \\ 0 & 0 & \ddots & 0 & 0 & \vdots \\ 0 & 0 & 0 & \multicolumn{2}{c|}{-I_2} & I_2 \end{array}\right] \begin{pmatrix} u_{01}^* \\ u_{11}^* \\ \vdots \\ u_{0J}^* \\ u_{1J}^* \\ \beta_0 \\ \beta_1 \end{pmatrix} + \left(\frac{e}{\delta}\right)$$

with

$$\Lambda = \begin{pmatrix} \sigma_e^2 I_N & 0 \\ 0 & \text{diag}_J(\Omega_u) \end{pmatrix}$$

Recall we now have:

$$p(\boldsymbol{\theta}|Y, X, \Lambda) \sim \text{MVN}\left((X^T\Lambda^{-1}X)^{-1}X^T\Lambda^{-1}Y, (X^T\Lambda^{-1}X)^{-1}\right)$$

where

$$X^T\Lambda^{-1}X = \left[\begin{array}{ccc|c} A_1 & 0 & 0 & -\Omega_u^{-1} \\ 0 & \ddots & 0 & \vdots \\ 0 & 0 & A_J & -\Omega_u^{-1} \\ \hline -\Omega_u^{-1} & \cdots & -\Omega_u^{-1} & J\Omega_u^{-1} \end{array}\right]$$

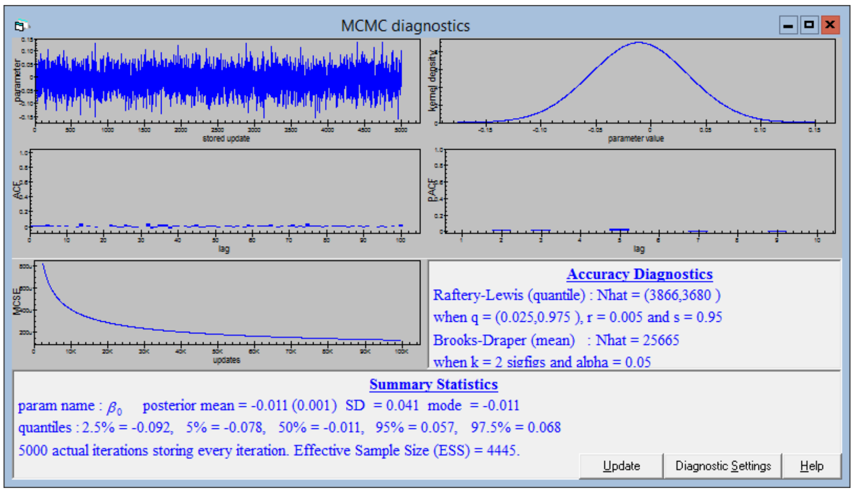
where $A_i = \Omega_u^{-1} + \frac{X_i^T X_i}{\sigma_e^2}$. Note that this makes the top left partition block diagonal and hence the technique to speed up matrix inversions discussed earlier can again be used. We will now look at how this works in practice.

21.6 Random Slopes model practice

Following on from the random intercepts model (after perhaps unselecting the **store residuals** option if you wish) we will now specify the random slopes model:

- Click on the **IGLS/RIGLS** tab on the **Estimation control** window.
- Click on **standlrt** in the **Equations** window.
- Click on the **j(school)** checkbox in the window that appears.
- Click on the **Done** button.
- Click **Start**.
- Select the tab labelled **MCMC** on the **Estimation control** window.
- Change the length of **monitoring chain** to 5000.
- Click **Start**.

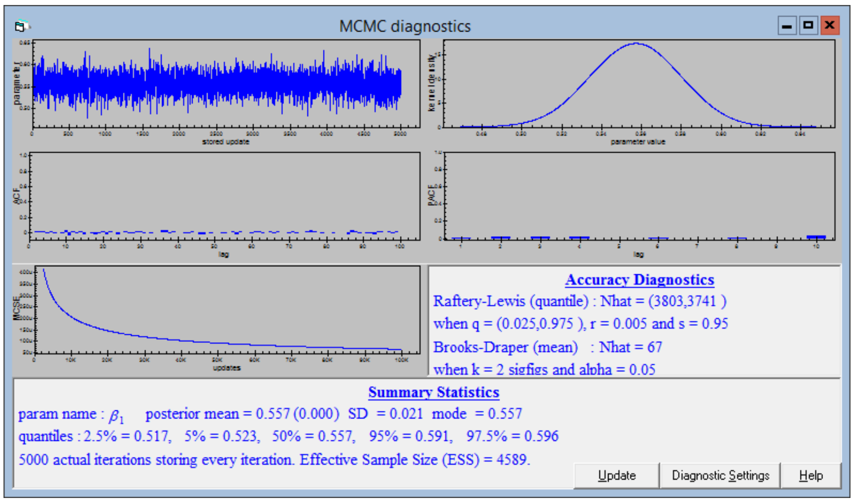
After running the random slopes model we can view the chains for the parameters by selecting the appropriate parameters from the **Trajectories** window. If we firstly look at the intercept parameter, β_0 we see the following:



Accuracy Diagnostics
Raftery-Lewis (quantile) : Nhat = (3866,3680)
when q = (0.025,0.975), r = 0.005 and s = 0.95
Brooks-Draper (mean) : Nhat = 25665
when k = 2 sigfigs and alpha = 0.05

Summary Statistics
param name : β_0 posterior mean = -0.011 (0.001) SD = 0.041 mode = -0.011
quantiles : 2.5% = -0.092, 5% = -0.078, 50% = -0.011, 95% = 0.057, 97.5% = 0.068
5000 actual iterations storing every iteration. Effective Sample Size (ESS) = 4445.

Here we have almost independent sampling once again with an effective sample size of 4445 iterations which compares with 281 for the standard algorithm. Similarly if we look at the average **standlrt** effect, β_1, we see the following:



Accuracy Diagnostics
Raftery-Lewis (quantile) : Nhat = (3803,3741)
when q = (0.025,0.975), r = 0.005 and s = 0.95
Brooks-Draper (mean) : Nhat = 67
when k = 2 sigfigs and alpha = 0.05

Summary Statistics
param name : β_1 posterior mean = 0.557 (0.000) SD = 0.021 mode = 0.557
quantiles : 2.5% = 0.517, 5% = 0.523, 50% = 0.557, 95% = 0.591, 97.5% = 0.596
5000 actual iterations storing every iteration. Effective Sample Size (ESS) = 4589.

Here we again have good mixing and the ESS of 4589 compares with 806 for the standard algorithm. Note here that fitting random slopes tends to worsen the mixing for the specific fixed effect in the standard algorithm. So we see that once again SMCMC is much better than the standard Gibbs sampling algorithm. As always there is a computational overhead, in this case it takes 15.5 seconds to run SMCMC as opposed to 5.5 seconds using the standard algorithm. It is worth considering that the **tutorial** dataset has only a small number of level 2 units (65) and the cost in terms of speed of SMCMC will increase with larger numbers of level 2 units. So although on the evidence of this chapter, SMCMC looks to be a great advance, we will demonstrate in later chapters other methods that also improve on the standard algorithm but without the computational overhead.

Chapter learning outcomes

- ⋆ How to use structured MCMC methods for Normal data
- ⋆ How SMCMC improves both fixed effect and residual chains
- ⋆ How to use SMCMC with random slopes
- ⋆ The advantages and disadvantages of SMCMC

Chapter 22

Using the Structured MVN framework for models

In this chapter, as with the last chapter, we introduce some new MCMC methods that are limited to a subset of the models that can be fitted in MLwiN. The methods described in this chapter are described in more detail in Browne et al. (2009*a*) where several models are considered using standalone code. We have however only implemented 2-level variance components models in MLwiN following the findings in Browne et al. (2009*a*).

We will firstly give some theoretical background in the next section to the ideas behind the Structured MVN framework of models. For the reader who is less interested in the theory, as long as they consider the methods simply as another alternative fast method for fitting variance components models in MCMC then they can skip forward to the practical sections that follow.

22.1 MCMC theory for Structured MVN models

When fitting multilevel models we are assuming some structure exists in the data and often this amounts to clustering in the dataset where certain observations are collected from the same level 2 unit and it is believed that such observations should be more similar than observations collected from different level 2 units.

There are (at least) two ways of describing a multilevel model and we illustrate this here for a Normal response variance components model with response vector y, predictor variables X and data collected on J level 2 units with n_j observations in the jth level 2 unit.

Firstly the random effect (latent variable) model, which is the form we have considered in earlier chapters, describes the structure by adding random (unobserved) effects, u_j to a standard linear model so that we have the following:

$$\begin{aligned} y_{ij} &= X_{ij}\boldsymbol{\beta} + u_j + e_{ij} \\ u_j &\sim \mathrm{N}(0, \sigma_u^2), \quad i = 1, \ldots, n_j, \\ e_{ij} &\sim \mathrm{N}(0, \sigma_e^2), \quad i = 1, \ldots, n_j, \quad j = 1, \ldots, J \end{aligned} \tag{22.1}$$

Here i indexes the observations in level 2 unit j and so y_{ij} is the ith response on the jth level 2 unit, $\boldsymbol{\beta}$ is the fixed effects vector associated with the predictors X and u_j are the random effects, one for each level 2 unit.

As we have seen in earlier chapters, assuming conjugate priors are used then this model can be fitted using a four step Gibbs sampling algorithm.

An alternative formulation is to consider the whole response vector y as following a multivariate Normal distribution as follows:

$$y \sim \mathrm{MVN}(X\boldsymbol{\beta}, V)$$

The clustering can now be accounted for by choosing the form given to V. We will describe such a model formulation as a structured multivariate Normal (SMVN) model. A simple linear model can be described in this framework by letting $V = \sigma_e^2 I_N$ and in the random intercepts case given above we can write

$$\begin{aligned} y_j &\sim \mathrm{MVN}(X_j\boldsymbol{\beta}, V_j) \\ V_j &= \sigma_e^2 I_{n_j} + \sigma_u^2 J_{n_j} \end{aligned} \tag{22.2}$$

where y_j is the response vector for the jth level 2 unit, X_j is the matrix of p predictor variables for the jth level 2 unit, I_{n_j} is the $n_j \times n_j$ identity matrix and J_{n_j} is an $n_j \times n_j$ matrix of 1s. This formulation is the basis of the IGLS algorithm that is the default estimation method used in MLwiN. It is a straightforward exercise to verify that the two formulations result in the same model.

We will in this chapter consider how to construct an MCMC algorithm for the second formulation of the multilevel model. One feature of this formulation is the lack of the random effects, u_j, and consequently we have 3 sets of unknown parameters, the fixed effects $\boldsymbol{\beta}$, the level 2 variance σ_u^2, and the level 1 variance σ_e^2. All these parameters require prior distributions and so for generality we will assume generic priors $p(\boldsymbol{\beta}), p(\sigma_u^2)$ and $p(\sigma_e^2)$.

We then combine these with the likelihood to form the posterior distribution. The likelihood is as follows:

$$\mathrm{L}(y \mid \boldsymbol{\beta}, \sigma^2_u, \sigma^2_e) = \textstyle\prod_{j=1}^{J} (2\pi)^{-n_j/2} |V_j| \exp\left[-\frac{1}{2}(y_j - X_j\boldsymbol{\beta})^T V_j^{-1}(y_j - X_j\boldsymbol{\beta})\right]$$
$$\text{where } V_j = \sigma^2_e I_{n_j} + \sigma^2_u J_{n_j}.$$

This likelihood (and loglikelihood) can be simplified for special cases (as illustrated in McCulloch & Searle, 2001), for example, for a general random intercept model the loglikelihood is

$$\begin{aligned}\log(\mathrm{L}(y \mid \boldsymbol{\beta}, \sigma^2_u, \sigma^2_e)) = -\tfrac{1}{2}N\log 2\pi - \tfrac{1}{2}\textstyle\sum_j \log(\sigma^2_e + n_j\sigma^2_u) - \tfrac{1}{2}(N-J)\log(\sigma^2_e)\\ -\tfrac{1}{2\sigma^2_e}\textstyle\sum_{i,j}(y_{ij} - X_{ij}\boldsymbol{\beta})^2 + \tfrac{\sigma^2_u}{2\sigma^2_e}\textstyle\sum_j \frac{(\sum_i (y_{ij} - X_{ij}\boldsymbol{\beta}))^2}{\sigma^2_e + n_j\sigma^2_u}.\end{aligned}$$

Given the above we can construct a three step Metropolis Hastings algorithm (using suitable starting values):

Step 1: Update the fixed effects $\boldsymbol{\beta}$ using univariate random walk Metropolis at iteration $t+1$ as follows, for $j = 1, \ldots, p$ and with $\boldsymbol{\beta}_{(-j)}$ signifying the $\boldsymbol{\beta}$ vector without component j:

$$\begin{aligned}\beta_j(t+1) &= \beta_j(*) \text{ with probability } \min\left[1, \tfrac{p(\beta_j(*) \mid y, \sigma^2_u, \sigma^2_e, \boldsymbol{\beta}_{(-j)})}{p(\beta_j(t) \mid y, \sigma^2_u, \sigma^2_e, \boldsymbol{\beta}_{(-j)})}\right]\\ &= \beta_j(t) \text{ otherwise,}\end{aligned}$$

where $\beta_j(*) \sim \mathrm{N}(\beta_j(t), s^2_{\beta j})$ and $p(\beta_j \mid y, \sigma^2_u, \sigma^2_e, \boldsymbol{\beta}_{(-j)}) \propto L(y \mid \boldsymbol{\beta}, \sigma^2_u, \sigma^2_e)p(\beta_j)$.

Step 2: Update σ^2_u using univariate random walk Metropolis at iteration $t+1$ as follows :

$$\begin{aligned}\sigma^2_u(t+1) &= \sigma^2_u(*) \text{ with probability } \min\left[1, \tfrac{p(\sigma^2_u(*) \mid y, \boldsymbol{\beta}, \sigma^2_e)}{p(\sigma^2_u(t) \mid y, \boldsymbol{\beta}, \sigma^2_e)}\right]\\ &= \sigma^2_u(t) \text{ otherwise,}\end{aligned}$$

where $\sigma^2_u(*) \sim \mathrm{N}(\sigma^2_u(t), s^2_{pu})$ and $p(\sigma^2_u \mid y, \boldsymbol{\beta}, \sigma^2_e) \propto \mathrm{L}(y \mid \boldsymbol{\beta}, \sigma^2_u, \sigma^2_e)p(\sigma^2_u)$.

Here as described previously in Browne (2006) we use a Normal proposal distribution and assume that inadmissible values for σ^2_u will have zero likelihood or a prior of zero.

Step 3: Update σ^2_e using univariate random walk Metropolis at iteration $t+1$ as follows :

$$\begin{aligned}\sigma^2_e(t+1) &= \sigma^2_e(*) \text{ with probability } \min\left[1, \tfrac{p(\sigma^2_e(*) \mid y, \boldsymbol{\beta}, \sigma^2_u)}{p(\sigma^2_e(t) \mid y, \boldsymbol{\beta}, \sigma^2_u)}\right]\\ &= \sigma^2_e(t) \text{ otherwise,}\end{aligned}$$

where $\sigma_e^2(*) \sim \mathrm{N}(\sigma_e^2(t), s_{pe}^2)$ and $p(\sigma_e^2 \mid y, \boldsymbol{\beta}, \sigma_u^2) \propto \mathrm{L}(y \mid \boldsymbol{\beta}, \sigma_u^2, \sigma_e^2) p(\sigma_e^2)$.

To choose the values of the proposal standard deviations, $s_{\beta j}, s_{pu}$ and s_{pe} we use the adapting procedure from Browne & Draper (2000) described in earlier chapters.

To speed up the algorithm it is useful to consider how most efficiently to evaluate the likelihood. In fact it is easier to work with the log-likelihood than the likelihood. For the general variance components model we have

$$\log(\mathrm{L}(y \mid \boldsymbol{\beta}, \sigma_u^2, \sigma_e^2)) = -\tfrac{1}{2}N\log 2\pi - \tfrac{1}{2}\sum_j \log(\sigma_e^2 + n_j\sigma_u^2) - \tfrac{1}{2}(N-J)\log(\sigma_e^2)$$
$$-\tfrac{1}{2\sigma_e^2}\sum_{i,j}(y_{ij} - X_{ij}\boldsymbol{\beta})^2 + \tfrac{\sigma_u^2}{2\sigma_e^2}\sum_j \tfrac{(\sum_i (y_{ij} - X_{ij}\boldsymbol{\beta}))^2}{\sigma_e^2 + n_j\sigma_u^2}.$$

Here the majority of the computation lies in the last two terms with their summations of squared terms over the length of the dataset. We can speed up evaluation by considering summary statistics of the data that are constant through iterations.

If we write y for the vector of responses y_{ij} and X for the matrix of predictor variables then we can write the penultimate term, $\sum_{i,j}(y_{ij} - X_{ij}\boldsymbol{\beta})^2 = y^T y - 2y^T X\boldsymbol{\beta} + \boldsymbol{\beta}^T X^T X\boldsymbol{\beta}$. If we were to store the sums of squares and cross product objects $y^T y, y^T X$ and $X^T X$ then we would need to do far fewer numerical operations at each iteration. We could also construct the same objects for each cluster and the operations for the inner i summation in the final term can also be reduced. These efficiency gains rely on the fact that the number of observations in the dataset N (and the size of each cluster n_j) is far greater than the number of fixed effects p which is true in most cases.

We will next look at how we run this algorithm in practice.

22.2 Using the SMVN framework in practice

As with the last chapter we will investigate the **tutorial** dataset for comparison with the standard methods used in the earlier chapters. Firstly we need to retrieve the worksheet:

- Select **Open sample worksheet** from the **File** menu.
- Select **tutorial.ws**.

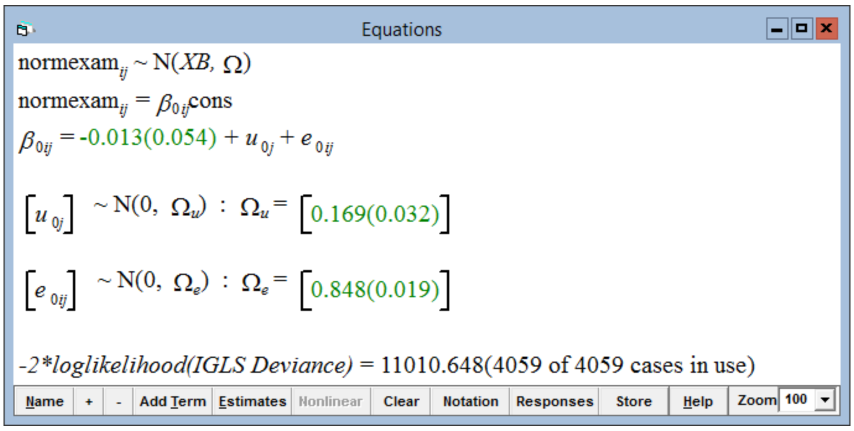
This will open the **Names** window and we now need once again to bring up the **Equations** window and set up the model. Here again we will set up a model with no predictor variables

- Select **Equations** from the **Model** menu.
- Click on **y** (either of the **y** symbols shown will do).
- In the **y** list, select **normexam**.
- In the **N levels** list, select **2-ij**.
- In the **level 2(j):** list, select **school**.
- In the **level 1(i):** list, select **student**.
- Click on the **done** button.
- Click on the red x_0.
- In the drop-down list, select **cons**.
- Check the box labelled **i(student)**.
- Check the box labelled **j(school)**.
- Click on the **Done** button.

These commands will set up the model which we now need to fit using MCMC, although firstly as usual we will fit it using IGLS to get starting values.

- Click **Start**.

The **Equations** should now look as follows:



$\text{normexam}_{ij} \sim \text{N}(XB, \Omega)$

$\text{normexam}_{ij} = \beta_{0ij}\text{cons}$

$\beta_{0ij} = -0.013(0.054) + u_{0j} + e_{0ij}$

$\begin{bmatrix} u_{0j} \end{bmatrix} \sim \text{N}(0, \Omega_u) : \Omega_u = \begin{bmatrix} 0.169(0.032) \end{bmatrix}$

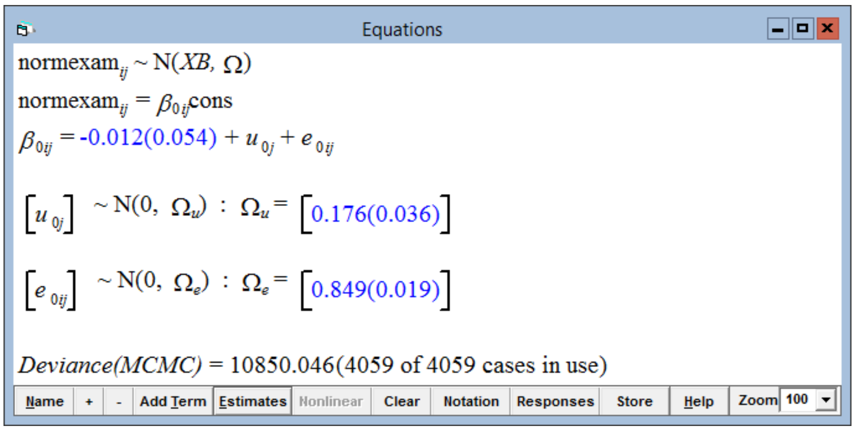
$\begin{bmatrix} e_{0ij} \end{bmatrix} \sim \text{N}(0, \Omega_e) : \Omega_e = \begin{bmatrix} 0.848(0.019) \end{bmatrix}$

*-2*loglikelihood(IGLS Deviance)* = 11010.648(4059 of 4059 cases in use)

Here we see the deviance calculated by IGLS is 11010.648. We will firstly run this model using the standard MCMC algorithm that assumes a random effect formulation.

- Click on the **Estimation Control** button
- Select the tab labelled **MCMC**
- Click **Start**.

Upon running the model the **Equations** window should look as follows:



$\text{normexam}_{ij} \sim \text{N}(XB, \Omega)$

$\text{normexam}_{ij} = \beta_{0ij}\text{cons}$

$\beta_{0ij} = -0.012(0.054) + u_{0j} + e_{0ij}$

$\left[u_{0j}\right] \sim \text{N}(0, \Omega_u) : \Omega_u = \left[0.176(0.036)\right]$

$\left[e_{0ij}\right] \sim \text{N}(0, \Omega_e) : \Omega_e = \left[0.849(0.019)\right]$

Deviance(MCMC) = 10850.046(4059 of 4059 cases in use)

Here we see that the Deviance is 10850.046 which is very different from that given by IGLS. This is because IGLS is using a multivariate Normal model likelihood whilst the MCMC method is using the product of univariate Normal likelihoods conditional on the random effects.

To calculate the DIC diagnostic for our model:

- Select **MCMC/DIC diagnostic** from the **Model** menu.

This will bring up the **Output** window with the following information:

Dbar	D(thetabar)	pD	DIC
10850.05	10790.01	60.03	10910.08

Here as described earlier we have 60 'effective' parameters as the random effects are shrunk due to sharing a prior distribution and so do not contribute whole parameters to the parameter count.

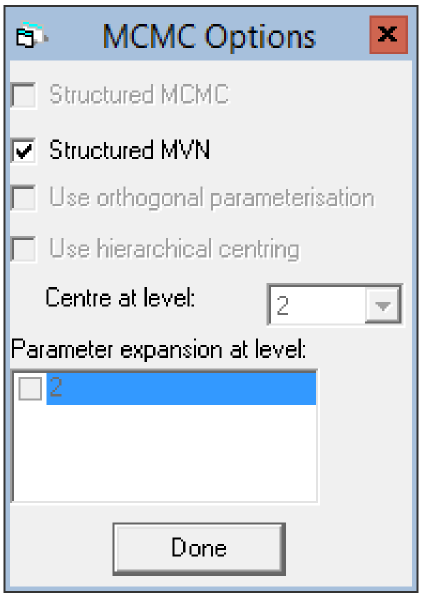
We have previously looked at the diagnostics for the various parameters in this model and in particular via the **Trajectories** window we can get the effective sample size for each parameter in the model. These are 191, 3305 and 4812 for β_0, σ^2_{u0} and σ^2_{e0} respectively.

We will now consider how to run this model using the structured MVN formulation.

- Click on the **IGLS/RIGLS** tab on the **Estimation control** window.
- Click **Start**.

- Click on the **Estimation Control** button
- Select the tab labelled **MCMC** on the **Estimation control** window.
- Select **MCMC/MCMC options** from the **Model** menu.
- Click on **Structured MVN**

The **MCMC options** window should then look as follows:

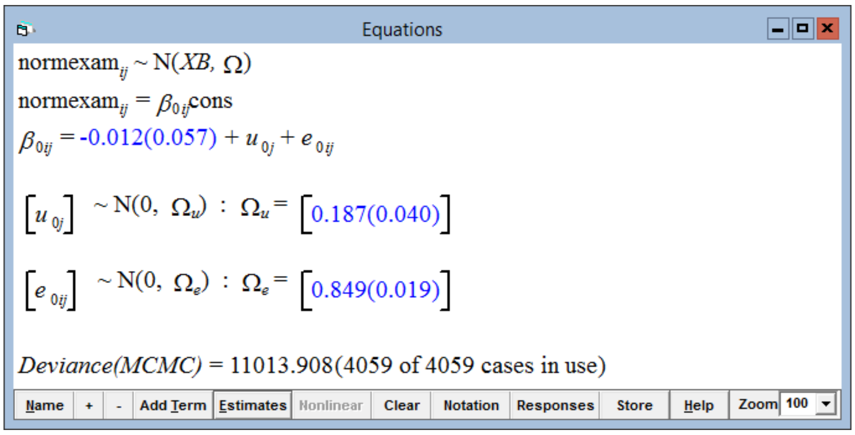

MCMC Options
Structured MCMC
Structured MVN
Use orthogonal parameterisation
Use hierarchical centring
Centre at level:
2
Parameter expansion at level:
2
Done

We will next run the model using the SMVN formulation by doing the following:

- Click on the **Done** button on the **MCMC options** window.
- Click **Start**.

This method is far quicker than the standard MCMC algorithm and if you close all the windows while running the algorithm it will finish in less than a second.

The **Equations** window should now look as follows:



$\text{normexam}_{ij} \sim \text{N}(XB, \Omega)$

$\text{normexam}_{ij} = \beta_{0ij}\text{cons}$

$\beta_{0ij} = -0.012(0.057) + u_{0j} + e_{0ij}$

$\left[u_{0j}\right] \sim \text{N}(0, \Omega_u) : \Omega_u = \left[0.187(0.040)\right]$

$\left[e_{0ij}\right] \sim \text{N}(0, \Omega_e) : \Omega_e = \left[0.849(0.019)\right]$

Deviance(MCMC) = 11013.908(4059 of 4059 cases in use)

One first thing to note is that the level 2 variance estimate σ^2_{u0} here is larger than seen for the random effects model. This is because the SMVN coding in MLwiN has only been done for uniform priors and so any priors displayed on the **Equations** window should be ignored. A uniform prior for the variance will give a larger posterior mean estimate than a $\Gamma^{-1}(\varepsilon, \varepsilon)$ prior as has been discussed in earlier chapters.

Here we see the deviance calculated by MCMC is 11013.908 which is much closer to that from IGLS (11010.648). Calculating the DIC diagnostic for our model we now get

Dbar	D(thetabar)	pD	DIC
11013.91	11010.92	2.98	11016.89

The main thing to note here is that the deviance given in the **Equations** window is the mean deviance (11013.91) rather than the deviance at the mean (11010.92) which is closely related to the IGLS deviance (11010.65) which is effectively the deviance at the mode. The other thing to note is that p_D is given as 2.98, effectively 3 which corresponds to the number of actual parameters in the model as we no longer have random effects.

If we again look at diagnostics for the various parameters in the model and in particular via the **Trajectories** window we look at the ESS for each parameter in the model we get 1182, 974 and 1201 for β_0, σ^2_{u0} and σ^2_{e0} respectively. Here we see an improvement for β_0 as we no longer have the problem of correlation between it and the residuals. All 3 parameters have reasonably similar ESS and these are smaller than the actual run-length primarily due to using Metropolis sampling. However the methods are incredibly fast and so the time to get a specific ESS will be competitive if not better than standard methods for all parameters.

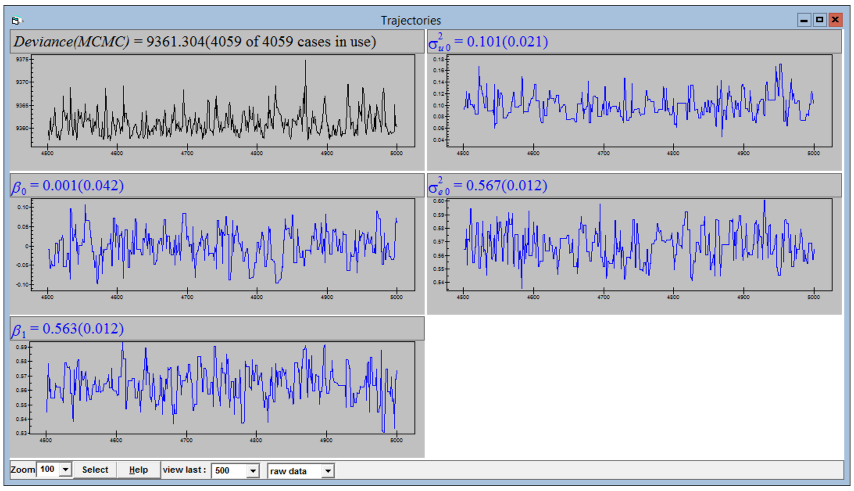
We next look at how we can perform model comparison with other predictor variables before considering how to check if clustering is required.

22.3 Model Comparison and structured MVN models

We will next investigate how in this framework we can choose between models by adding the predictor variable **standlrt** to the model. We can also compare how the SMVN framework compares with both the standard method and SMCMC when we add in this predictor to the model. Firstly we need to set up the model:

- Click on the **IGLS/RIGLS** tab on the **Estimation control** window.
- Click the **Add Term** button on the **Equations** window tool bar.
- Select **standlrt** from the **variable** list.
- Click on the **Done** button.
- Click **Start**.
- Select the tab labelled **MCMC** on the **Estimation control** window.
- Click **Start**.

After the model runs the **Trajectories** window will look as follows:


Trajectories
Deviance(MCMC) = 9361.304(4059 of 4059 cases in use)
σ^2_{u0} = 0.101(0.021)
β_0 = 0.001(0.042)
σ^2_{e0} = 0.567(0.012)
β_1 = 0.563(0.012)
Zoom 100
Select
Help
view last : 500
raw data

Once again the estimation is almost instantaneous with most of the time taken in refreshing windows. The effective sample sizes can be compared with the standard Gibbs sampler and structured MCMC in the table below. Here again we see that the method produces smaller ESS scores in general, however these should be balanced by the faster computations.

Parameter	Gibbs	SMCMC	SMVN
β_0	216	5209	1125
β_1	4413	5332	1219
σ^2_{u0}	2821	3108	1015
σ^2_{e0}	4712	4707	1276

If we look at the DIC diagnostic for this model via the **Model** menu we see

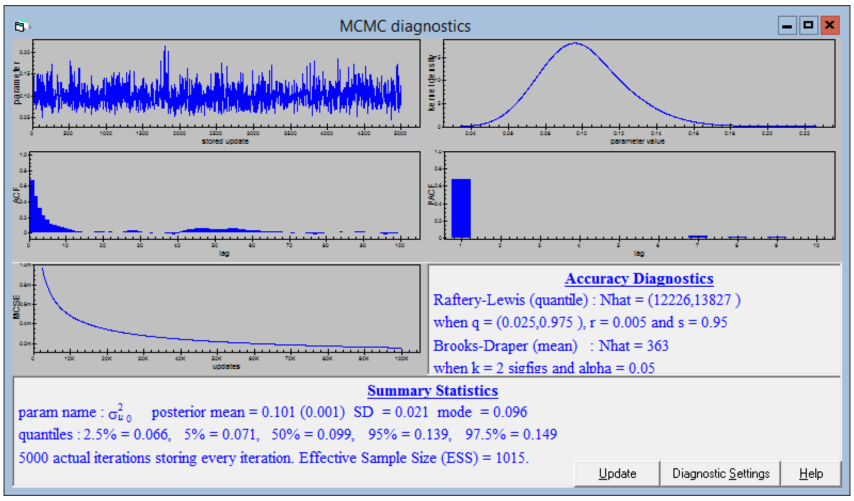
Dbar	D(thetabar)	pD	DIC
9361.30	9357.45	3.86	9365.16

Here again p_D gives roughly the actual number of parameters in the model and as seen for the standard Gibbs sampling algorithm we have a greatly improved model. Again the deviance at the mean (9357.45) is close to the IGLS deviance (9357.242) and hence the change in deviance is also similar (1653.47 versus 1653.41) although this does not mean we should use it in a likelihood ratio test.

22.4 Assessing the need for the level 2 variance

One problem when fitting models using MCMC is assessing whether a set of random effects should be included in the model. When testing a fixed effect we have a myriad of options ranging from looking at the change in DIC between models to examining the credible interval for the additional parameter for the presence of zero in the interval. We cannot do the same thing with a variance parameter for the set of random effects as we are restricted by the fact that a variance is typically assumed positive and so the prior used generally has no support for negative values. In the structured MVN framework, however, the level 2 variance parameter σ^2_u simply has a role as a parameter in the global variance matrix V. It is in this framework perfectly plausible for the parameter to take negative values to represent less correlation within a cluster than is seen from a randomly selected group from the population at large and a small negative correlation will still result in a positive definite V matrix.

Such negative correlations are seen in practice: for example, in clusters of animal litters where competition for a shared resource tends to produce large variation within a cluster. For the **tutorial** example we see, via the **MCMC diagnostics** window, that there is no support for a negative value for σ^2_u:



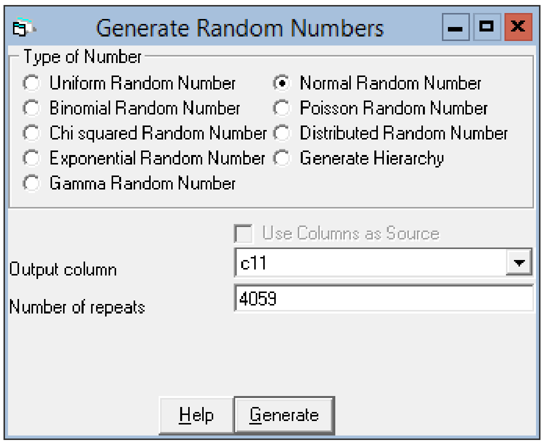
Accuracy Diagnostics
Raftery-Lewis (quantile) : Nhat = (12226,13827)
when q = (0.025,0.975), r = 0.005 and s = 0.95
Brooks-Draper (mean) : Nhat = 363
when k = 2 sigfigs and alpha = 0.05

Summary Statistics
param name : σ^2_{u0} posterior mean = 0.101 (0.001) SD = 0.021 mode = 0.096
quantiles : 2.5% = 0.066, 5% = 0.071, 50% = 0.099, 95% = 0.139, 97.5% = 0.149
5000 actual iterations storing every iteration. Effective Sample Size (ESS) = 1015.

We can however consider a model where we know that there is no real level 2 variation. To do this we will simulate a response variable using the random number generator in MLwiN.

- Select **Generate Random Numbers** from the **Basic Statistics** menu.
- Click the **Normal Random Number** button on the **Generate Random Numbers** window.
- Select column **C11** from the **output column** list.
- Input **4059** into the **Number of repeats** box.

The window should now look as follows:



Type of Number
Uniform Random Number [✓ Normal Random Number]
Binomial Random Number Poisson Random Number
Chi squared Random Number Distributed Random Number
Exponential Random Number Generate Hierarchy
Gamma Random Number

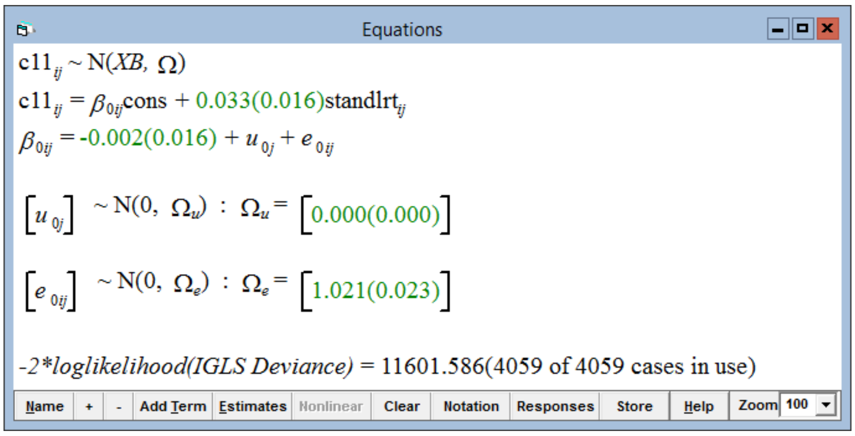
Use Columns as Source
Output column c11
Number of repeats 4059

We will next generate this column and use it as a response in an IGLS run of the model

- Click the **Generate** button on the **Generate Random Numbers** window.

- Click on the **IGLS/RIGLS** tab on the **Estimation control** window.
- Click on the response name (**normexam**$_{ij}$) in the **Equations** window.
- Select **C11** from the **y variable** list.
- Click on the **Done** button.
- Click **Start**.

This will fit the previously considered variance components model but with our simulated response using IGLS. The results can be seen below:



$c11_{ij} \sim N(XB, \Omega)$

$c11_{ij} = \beta_{0ij}\text{cons} + 0.033(0.016)\text{standlrt}_{ij}$

$\beta_{0ij} = -0.002(0.016) + u_{0j} + e_{0ij}$

$\begin{bmatrix} u_{0j} \end{bmatrix} \sim N(0,\ \Omega_u) \ : \ \Omega_u = \begin{bmatrix} 0.000(0.000) \end{bmatrix}$

$\begin{bmatrix} e_{0ij} \end{bmatrix} \sim N(0,\ \Omega_e) \ : \ \Omega_e = \begin{bmatrix} 1.021(0.023) \end{bmatrix}$

*-2*loglikelihood(IGLS Deviance)* = 11601.586(4059 of 4059 cases in use)

The IGLS algorithm gives a zero estimate for the level 2 variance - basically a check is included in the algorithm so that any negative estimates are reset to zero. This resetting can be removed via the **Estimation control** window although we do not demonstrate this here.

We will next run the model using MCMC in the structured MVN framework:

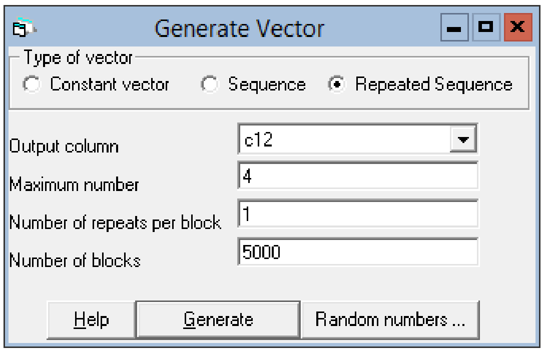
- Select the tab labelled **MCMC** on the **Estimation control** window.
- Click **Start**.

We could now look at the chain for the level 2 variance via the **Trajectories** window however this window will give a Kernel density plot restricted to positive values as the parameter is assumed to be positive. Instead we can take the data solely for the level 2 variance parameter and use the **Column Diagnostics** window. This is a slightly involved procedure as detailed below:

- Select **Generate Vector** from the **Data Manipulation** menu.
- Click the **Repeated Sequence** button on the **Generate Vector** window.

- Select column **C12** from the **output column** list.
- Input **4** into the **Maximum number** box.
- Input **1** into the **Number of repeats per blocks** box.
- Input **5000** into the **Number of blocks** box.

We are here creating an indicator vector which will match up the chains for the four stored parameters that are currently stacked in column c1090. The **Generate Vector** window should look as follows:


Generate Vector

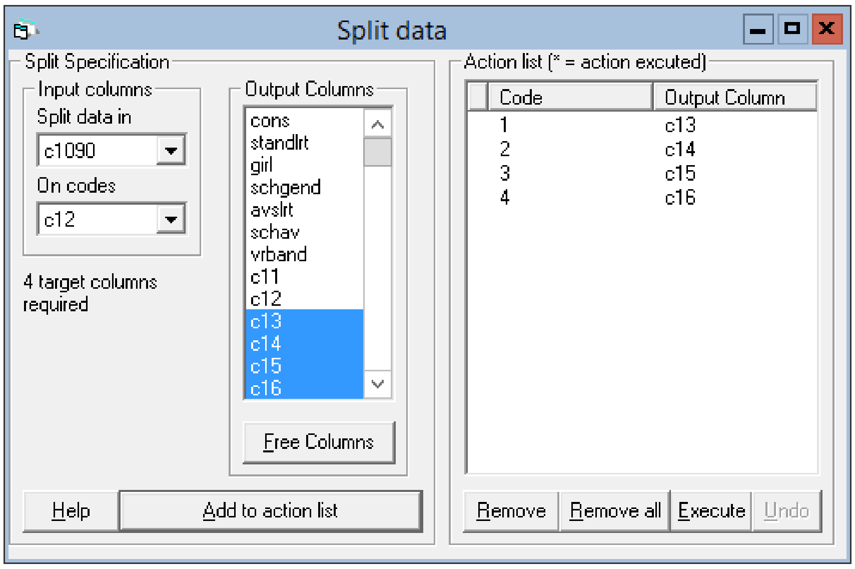
Type of vector
Constant vector Sequence [✓ Repeated Sequence]

Output column	c12
Maximum number	4
Number of repeats per block	1
Number of blocks	5000

We next need to use this vector to split up the various chains using the **Split column** window.

- Click the **Generate** button on the **Generate Vector** window.
- Select **Split column** from the **Data Manipulation** menu.
- Select column **C1090** from the **Split data in** list.
- Select column **C12** from the **On codes** list.
- Select columns **C13-C16** from the **Output Columns** list.
- Click the **Add to action list** button.

The **Split column** window will then look as follows:


Split data

Split Specification

Input columns

Split data in: c1090

On codes: c12

4 target columns required

Output Columns: cons, standlrt, girl, schgend, avslrt, schav, vrband, c11, c12, c13, c14, c15, c16

Free Columns

Help | Add to action list

Action list (* = action excuted)

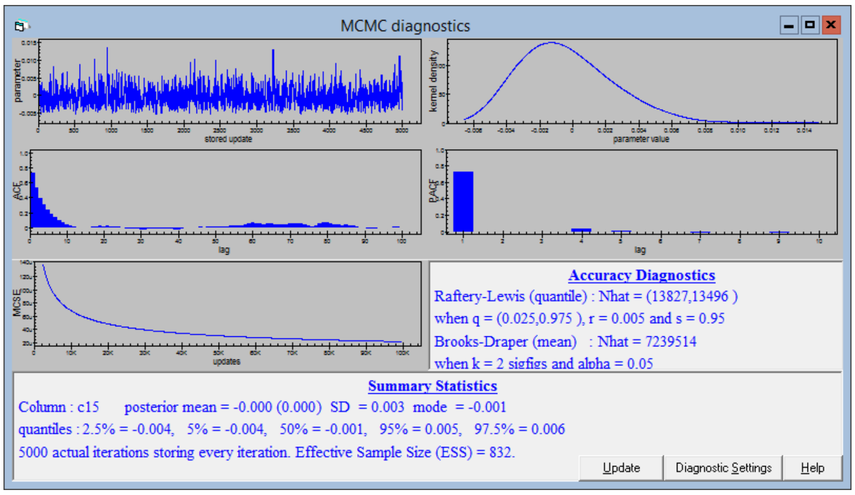
Code	Output Column
1	c13
2	c14
3	c15
4	c16

Remove | Remove all | Execute | Undo

We next need to perform the splitting operation and view the column (c15) that contains the level 2 variance chain.

- Click the **Execute** button on the **Split column** window.
- Select **Column Diagnostics** from the **Basic Statistics** menu.
- Select column **C15** from the **column** list.
- Click the **Apply** button.

The **MCMC Diagnostics** window will then look as follows:



Accuracy Diagnostics

Raftery-Lewis (quantile) : Nhat = (13827,13496)
when q = (0.025,0.975), r = 0.005 and s = 0.95
Brooks-Draper (mean) : Nhat = 7239514
when k = 2 sigfigs and alpha = 0.05

Summary Statistics

Column : c15 posterior mean = -0.000 (0.000) SD = 0.003 mode = -0.001
quantiles : 2.5% = -0.004, 5% = -0.004, 50% = -0.001, 95% = 0.005, 97.5% = 0.006
5000 actual iterations storing every iteration. Effective Sample Size (ESS) = 832.

Here we see that the parameter σ_u^2 represented by column 15 has a chain that contains both negative and positive estimates. We can now use the confidence interval constructed from the chain to see that there is no need to account for school effects in the model.

As a verification we can also look at the DIC diagnostic. The table below shows the DIC from this model fitted using the structured MVN framework and a simpler model with no random effects. Note to fit the simpler model using MCMC you have to switch off the Structured MVN option.

σ_u^2	Dbar	D(thetabar)	pD	DIC
With	11605.11	11601.44	3.67	11608.79
Without	11604.61	11601.59	3.02	11607.63

We see quite clearly that the DIC estimate is actually lower for the model without the random effect variance parameter included which backs up our earlier findings.

In this chapter we have introduced another MCMC method for fitting Normal response variance components models that links with the IGLS algorithm. The advantage with this method is its computational speed which more than compensates for the slightly smaller ESS values the method often produces. However the speed has to also be set against the scope of the method. Browne et al. (2009*a*) also investigated using the method for random slopes models and here the speed advantage disappears and so consequently the method has only been implemented in MLwiN for variance components methods.

Chapter learning outcomes

- ⋆ How to use the structured MVN framework with variance components models
- ⋆ How the DIC diagnostic works with these models
- ⋆ How to allow negative variances at level 2

Chapter 23

Using Orthogonal fixed effect vectors

In this chapter we will introduce the first of three reparameterisation methods that have been implemented in MLwiN. Basically a reparameterisation method involves reformulating a statistical model by replacing certain parameters with other parameters. This reparameterisation has to be done in such a way that it is possible to recover the original parameters in the model.

The first of these methods replaces a set of fixed effect predictors in a model with an alternative group of predictors that span the same parameter space but are orthogonal. We will give more details of what we mean by orthogonal vectors in the next section.

These methods have been implemented for all models in MLwiN and are universally useful, although in practice they are more useful for non-Normal response models. This is because the methods are best at removing the influence of correlation between chains for different fixed effects, and for Normal models all fixed effects are generally updated in a block.

In this chapter we revisit one model from each of three previous chapters and show the improvements that using orthogonal predictors produce. We would recommend ALWAYS using orthogonal predictors where possible for non-Normal models although we have not updated the material in earlier chapters. We will also investigate how the implementation of the methods has been included in the MLwiN to WinBUGS interface at the end of the chapter. For further details of the use of these methods and the other reparameterisation methods used in later chapters see Browne et al. (2009*b*). We now give some further details of how these methods work in practice.

23.1 A simple example

We will start here by considering a very simple example with some response variable y. Let us suppose we believe the value of y depends on the gender of the individual and so we have a regression model

$$y_i = \beta_0 + \beta_1 \mathbf{gender}_i + e_i, \quad e_i \sim \mathrm{N}(0, \sigma_e^2)$$

that contains just an intercept (β_0) and a gender vector which takes values 1 for girls and 0 for boys with associated coefficient β_1. Then β_0 would represent the average value of y for boys in the dataset and β_1 the difference between the averages for boys and girls. Now we could reparameterise this model as follows:

$$y_i = \beta_0(1 - \mathbf{gender}_i) + \beta_1^* \mathbf{gender}_i + e_i, \quad e_i \sim \mathrm{N}(0, \sigma_e^2)$$

Here we have replaced the intercept with a vector that takes values 1 for boys and 0 for girls and now β_0 still represents the average value of y for boys but β_1^* represents the average value of y for girls. We are able to recover the original β_1 as $\beta_1 = \beta_1^* - \beta_0$ and so this reparameterisation is simply a different way of expressing the same model and we would expect the same estimates for e_i and σ_e^2 no matter which parameterisation is used.

The second parameterisation has the additional property of containing orthogonal predictor variables. For a set of predictor variables to be orthogonal, the product of any pair of predictor vectors must be 0 i.e. $\sum_{i=1}^N x_{ji} x_{ki} = 0$ would mean predictors x_j and x_k are orthogonal.

In our example we have $x_{ji} = \mathbf{gender}_i$ and $x_{ki} = 1 - \mathbf{gender}_i$ which means that for every i either $x_{ji} = 0$ or $x_{ki} = 0$ and so their product is always zero and as a consequence the sum of all terms is 0. Note that this is a special case and the only property needed for orthogonality is that the product is 0. For the original formulation we had $x_{ji} = \mathbf{gender}_i$ and $x_{ki} = 1 \quad \forall i$ (the intercept) and so the product of the two vectors will equal the number of girls in the data which will never equal zero unless there are no girls in the dataset in which case we cannot identify a **gender** effect.

This example gives a good motivation of why orthogonal vectors are useful as clearly in this rather special case we could almost in fact, using the orthogonal parameterisation, split the data into two subsets (boys and girls) and then fit the models separately with the value of each parameter having no impact on the other. The only problem here is that both subsets would still share the same variance parameter whilst fitting the models separately would give two different estimates for σ_e^2 so it is not quite that simple. Clearly with a model that has the boys average and the difference between genders as parameters, changing one parameter will change the other i.e. if the boys' average increases then the difference between the two genders will decrease etc.

This is a special case but a similar property holds for orthogonal parameters in general: that the effect of each parameter should be independent of the others and so we should see better mixing of MCMC algorithms where the parameters are updated separately if an orthogonal parameterisation is used. Orthogonal parameterisations should also have benefits for likelihood based estimation procedures in improving convergence. We will next discuss how one constructs orthogonal vectors in general.

23.2 Constructing orthogonal vectors

Our general aim is given a set of predictor variable vectors $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N)$ to replace these vectors with an orthogonal set that spans the same predictor space. There are many such sets of orthogonal vectors and we will outline an algorithm that produces a unique set subject to the ordering of the predictors. The algorithm is as follows:

Step 1 : Number the predictors in some ordering $1, \ldots, N$.

Step 2 : Take each predictor in turn and replace it with a predictor that is orthogonal to all the (orthogonal) predictors already considered as described below:

For predictor $\mathbf{x}_k$ create $\mathbf{x}_k^* = w_{1,k}\mathbf{x}_1 + w_{2,k}\mathbf{x}_2 + \ldots w_{k-1,k}\mathbf{x}_{k-1} + \mathbf{x}_k$ so that $(\mathbf{x}_i^*)^T\mathbf{x}_k^* = 0 \quad \forall i < k$. Note that this results in solving $k-1$ equations in $k-1$ unknown w coefficients.

By performing this step for all predictors in turn we will end up with $\sum_{k=1}^{N} k - 1 = (N-1)N/2$ coefficients. Once we have performed step 2 for all predictors we have a lower diagonal matrix $\mathbf{W} = [w_{i,j}]$ such that $\mathbf{X}^* = \mathbf{W}\mathbf{X}$ and so we can run the model using the $\mathbf{X}^*$ predictors.

This will result in chains for parameters $\boldsymbol{\beta}^*$ which can be transformed into chains for the original parameters $\boldsymbol{\beta}$ by pre-multiplying $\boldsymbol{\beta}^*$ by $\mathbf{W}^T$. Each unique ordering of the predictors will give a unique set of orthogonal predictors. The algorithm will generally work provided the predictors are not co-linear.

From a Bayesian modelling point of view it is worth pointing out here that we would normally need to calculate the Jacobian of the transformation from $\boldsymbol{\beta}$ to $\boldsymbol{\beta}^*$ to ensure that we maintain the same prior distributions after reparameterisation. In all of our examples however we use improper uniform priors and so this is not a problem. It should however be noted that MLwiN will ignore any informative priors when orthogonalisation is switched on.

As we will see in the sections that follow, although it is useful to know how the algorithm works, the algorithm will simply run seamlessly in MLwiN without the user ever seeing the orthogonal vectors that are used but simply getting the output for the original parameterisation. We will now go through four examples showing specific models from earlier chapters. We will then compare the mixing of the chains between the standard and orthogonal methods.

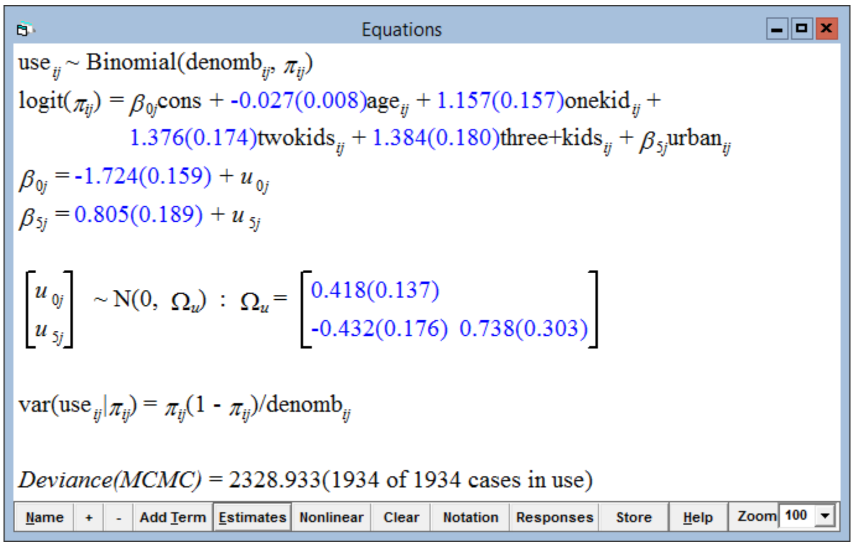
23.3 A Binomial response example

We will start by revisiting the Bangladeshi contraceptive use dataset which we studied in Chapter 10. In particular we will focus on the rather complicated example involving random coefficients for area type. We will now give instructions how to set up this model in MLwiN.

- Select **Open sample worksheet** from the **File** menu.
- Select **bang1.ws** from the list of worksheets.
- Select **Open**.
- Select **Equations** from the **Model** menu.
- Click on the **Clear** button to remove any existing model in the worksheet
- Click on the red y.
- Select **use** for the **y** variable.
- Select **2-ij** for the **number (N) of levels**.
- Select **district** as **level 2(j)**.
- Select **woman** as **level 1(i)**.
- Click on the **Done** button.
- Click on the **N** and instead choose **Binomial** from the list.
- Click on the **Done** button.
- Click on the red n_{ij}.
- Select **denomb** and click on the **Done** button.
- Click on the red x_0.
- Select **cons** and click on the **Done** button.
- Click on **cons** in the **Equations** window.
- Click on the **(j)district** box in the **X variable** window.

- Click on the **Done** button. (Note the level 1 variance is Binomial and so doesn't need adding).
- Click on the **Add Term** button.
- Select **age** from the **variable** list and click on the **Done** button.
- Click on the **Add Term** button on the **Equations** window.
- Select **lc** from the **variable** pull-down list.
- Click on the **Done** button.
- Click on the **Add Term** button on the **Equations** window.
- Choose **urban** from the **Variable** list.
- Click on the **Done** button.
- Click on the **urban** predictor.
- From the **X variable** window click in the **j(district)** tickbox.
- Click on the **Done** button.
- Click on the **Start** button.
- Change **Estimation Method** to **MCMC**.
- Click on the **Start** button.

After this rather long set of instructions the model will be run using the standard MCMC algorithm of Metropolis Sampling for the fixed effects and residuals and Gibbs sampling for the level 2 variance matrix. The **Equations** window will look as follows:



$$\text{use}_{ij} \sim \text{Binomial}(\text{denomb}_{ij}, \pi_{ij})$$
$$\text{logit}(\pi_{ij}) = \beta_{0j}\text{cons} + -0.027(0.008)\text{age}_{ij} + 1.157(0.157)\text{onekid}_{ij} + 1.376(0.174)\text{twokids}_{ij} + 1.384(0.180)\text{three+kids}_{ij} + \beta_{5j}\text{urban}_{ij}$$
$$\beta_{0j} = -1.724(0.159) + u_{0j}$$
$$\beta_{5j} = 0.805(0.189) + u_{5j}$$
$$\begin{bmatrix} u_{0j} \\ u_{5j} \end{bmatrix} \sim \text{N}(0,\ \Omega_u) \ : \ \Omega_u = \begin{bmatrix} 0.418(0.137) & \\ -0.432(0.176) & 0.738(0.303) \end{bmatrix}$$
$$\text{var}(\text{use}_{ij}|\pi_{ij}) = \pi_{ij}(1 - \pi_{ij})/\text{denomb}_{ij}$$

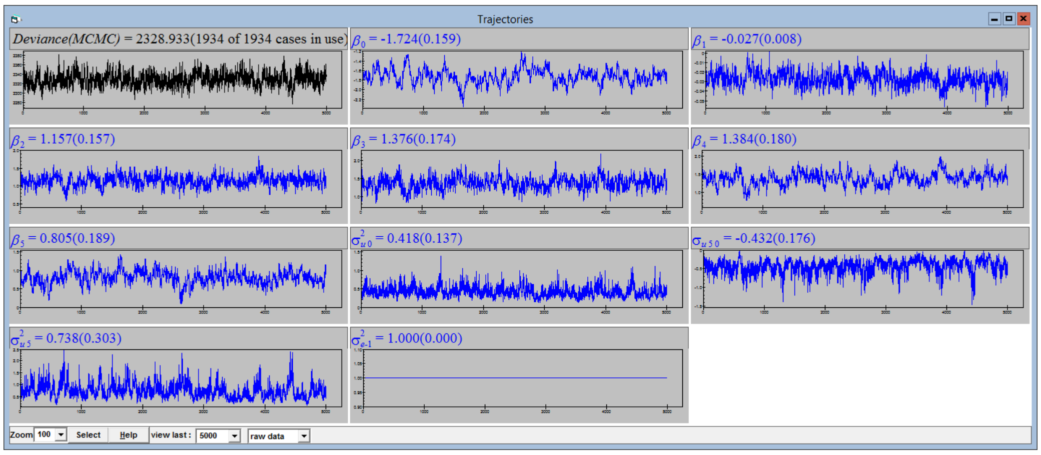
Deviance(MCMC) = 2328.933(1934 of 1934 cases in use)

We will need to look at the Trajectories window to examine the mixing of the parameters:

- Select **Trajectories** from the **Model** menu.

- Modify the **view last** box to **5000**.
- Click on the **select** button.
- In the pull down list choose **3 graphs per row**.
- Click on the **Done** button.

The **Trajectories** window will look as follows:

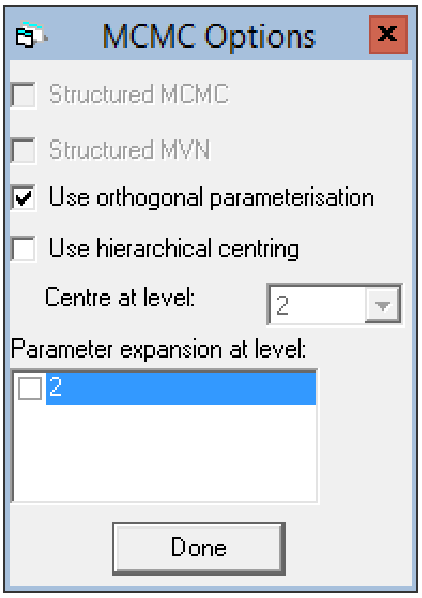

Trajectories
Deviance(MCMC) = 2328.933(1934 of 1934 cases in use)
β_0 = -1.724(0.159)
β_1 = -0.027(0.008)
β_2 = 1.157(0.157)
β_3 = 1.376(0.174)
β_4 = 1.384(0.180)
β_5 = 0.805(0.189)
σ^2_{u0} = 0.418(0.137)
σ_{u50} = -0.432(0.176)
σ^2_{u5} = 0.738(0.303)
σ^2_{e1} = 1.000(0.000)
Zoom 100
Select
Help
view last : 5000
raw data

Here we see that the mixing is not too bad overall however some parameters, for example β_0 display worse mixing than the rest. For each parameter we can click on the trace plot and get further diagnostics including effective sample sizes. We will tabulate and compare ESS values later.

We now need to rerun IGLS and choose to use the orthogonal option in MCMC.

- Click on the **IGLS/RIGLS** tab on the **Estimation control** window.
- Click **Start**.
- Click on the **Estimation Control** button
- Select the tab labelled **MCMC** on the **Estimation control** window.
- Select **MCMC/MCMC options** from the **Model** menu.
- Click on **Use orthogonal parameterisation**

The **MCMC options** window should then look as follows:


MCMC Options

- [ ] Structured MCMC
- [ ] Structured MVN
- [x] Use orthogonal parameterisation
- [ ] Use hierarchical centring

Centre at level: 2

Parameter expansion at level:

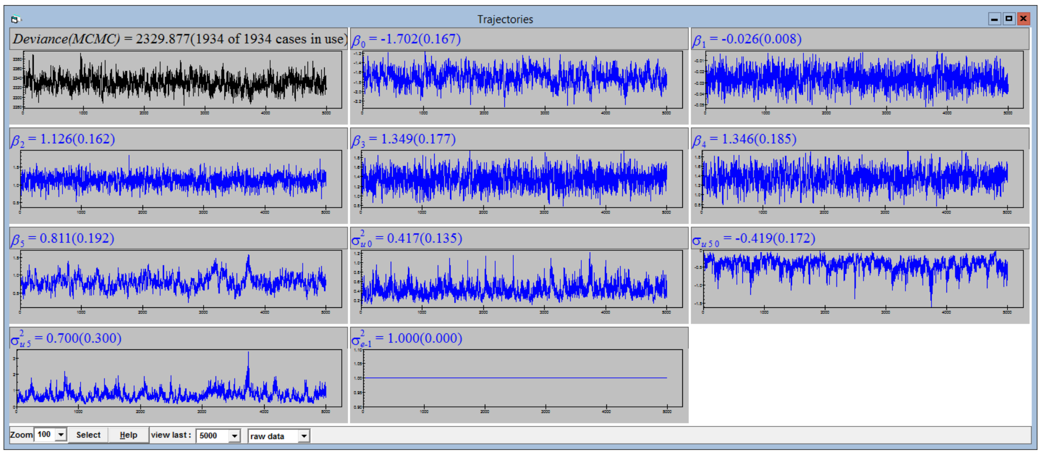
- [ ] 2

We will next run the model using the orthogonal parameterisation by doing the following:

- Click on the **Done** button on the **MCMC options** window.
- Click **Start**.

Note the method should not normally take any longer to run than the standard algorithm although with the **Trajectories** window in view and showing the last 5000 iterations this will slow estimation down. If you prefer you can close the **Trajectories** window and reopen it and reformat it as described earlier when estimation has completed.

Upon completing 5000 iterations the **Trajectories** window will look as follows:


Trajectories

Deviance(MCMC) = 2329.877(1934 of 1934 cases in use)

β_0 = -1.702(0.167)

β_1 = -0.026(0.008)

β_2 = 1.126(0.162)

β_3 = 1.349(0.177)

β_4 = 1.346(0.185)

β_5 = 0.811(0.192)

σ^2_{u0} = 0.417(0.135)

σ_{u50} = -0.419(0.172)

σ^2_{u5} = 0.700(0.300)

$\sigma^2_{e\text{-}1}$ = 1.000(0.000)

Zoom 100 | Select | Help | view last : 5000 | raw data

Even at first glance the improvement in mixing of the chains for all the fixed effects is impressive. In the table below we summarise the different ESS values for the two parameterisations.

Parameter	Non-orthogonal	Orthogonal
β_0	73	301
β_1	190	1154
β_2	183	978
β_3	143	1024
β_4	75	924
β_5	60	124
σ^2_{u0}	229	250
σ_{u50}	148	167
σ^2_{u5}	130	129

Here we see that the improvement for those fixed effects not also associated with a set of random effects is of the order 5-10 times larger an ESS. The two fixed effects β_0 and β_5 that are associated with sets of random effects show less (if any) improvement partly because their chains are also correlated with the chains for the sets of random effects. We will revisit this example in Chapter 25 when we investigate hierarchical centring to see if we can also improve the chains for these variables. The variance parameters should not be affected greatly by the orthogonal transformation and if anything for these parameters the situation is worse after the fixed effects have been orthogonalised.

Although some of the chains for this model mix poorly when orthogonal parameterisation is not used, we saw a Poisson model in Chapter 11 which exhibited far worse mixing and we investigate it next.

23.4 A Poisson example

In Chapter 11 we investigated using Poisson response models for a dataset that concerned Melanoma mortality in Europe. In section 11.4 we considered a model with fixed effects and interactions for nation and UV exposure and we will revisit this model again here. First we need to set up the model in MLwiN and hence we need to follow the following instructions.

- Select **Open sample worksheet** from the **File** menu.
- Select **mmmec.ws** and click on **Open**.
- Open the **Command interface** window from the **Data Manipulation** menu and enter the following commands:

```
► calc c9 = loge('exp')
► name c9 'logexp'
```

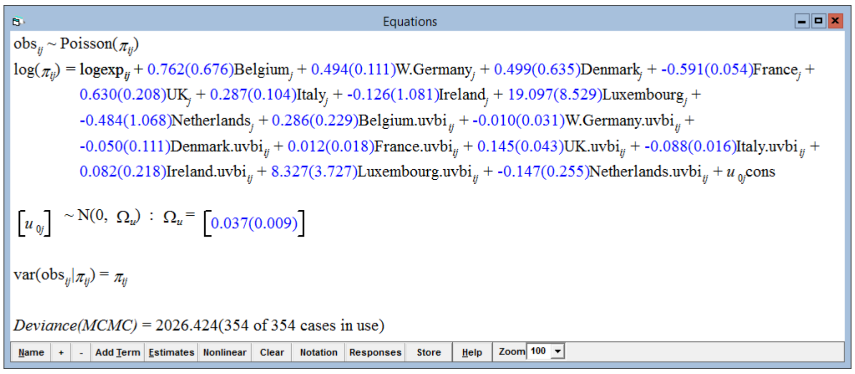
- Select **Equations** from the **Model** Menu.
- Click on the red y.
- Select **obs** as the y variable.
- Select **2-ij** as the number of levels.
- Select **region** as **level 2(j)**.
- Select **county** as **level 1(i)**.
- Click on the **Done** button.
- Click on the **N** and select **Poisson** from the popup list and click on the **Done** button.
- Click on the π_{ij} and from the **offset** window that appears select **logexp**.
- Click on the **Done** button.
- Click on the red x_0 and select **cons** and click on the **Done** button.
- In the **Equations** window click on β_0 (**cons**).
- Click in the **(j)region** tick box.
- Remove the tick in the **Fixed parameter** tick box.
- Click on the **Done** button.
- Click on the **Add Term** button on the **Equations** window.
- Select **nation** from the **variable** dropdown list.
- Select [**none**] from the **reference category** dropdown list.
- Click on the **Done** button.
- Click on the **Add Term** button on the **Equations** window.
- Select **1** in the **order** box.
- Select **nation** from the first **variable** dropdown list.
- Select **uvbi** from the second **variable** dropdown list.
- Select [**none**] from the first **ref cat** dropdown list.
- Click on the **Done** button.
- Click on the **Start** button.

The above list of instructions should result in setting up the model structure for the model in section 11.4 and fitting the model in IGLS. We next need to change estimation method to MCMC and repeat the model fitting of that section. We will also need to increase the run length because of the poor mixing of the chains for this model. This we do as follows:

- Click on the **Estimation Control** button.
- Click on the **MCMC** tab.

- Change the **Monitoring chain length** to 50,000.
- Change the **refresh rate** to 500.
- Click on the **Done** button.
- Click on the **Start** button.

Upon running the model we will get the following estimates:



$\text{obs}_{ij} \sim \text{Poisson}(\pi_{ij})$

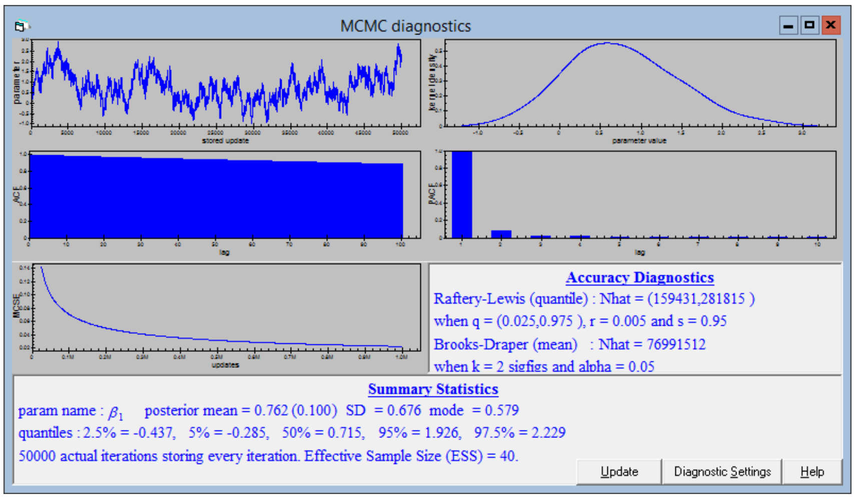
$\log(\pi_{ij}) = \text{logexp}_{ij} + 0.762(0.676)\text{Belgium}_j + 0.494(0.111)\text{W.Germany}_j + 0.499(0.635)\text{Denmark}_j + -0.591(0.054)\text{France}_j + 0.630(0.208)\text{UK}_j + 0.287(0.104)\text{Italy}_j + -0.126(1.081)\text{Ireland}_j + 19.097(8.529)\text{Luxembourg}_j + -0.484(1.068)\text{Netherlands}_j + 0.286(0.229)\text{Belgium.uvbi}_{ij} + -0.010(0.031)\text{W.Germany.uvbi}_{ij} + -0.050(0.111)\text{Denmark.uvbi}_{ij} + 0.012(0.018)\text{France.uvbi}_{ij} + 0.145(0.043)\text{UK.uvbi}_{ij} + -0.088(0.016)\text{Italy.uvbi}_{ij} + 0.082(0.218)\text{Ireland.uvbi}_{ij} + 8.327(3.727)\text{Luxembourg.uvbi}_{ij} + -0.147(0.255)\text{Netherlands.uvbi}_{ij} + u_{0j}\text{cons}$

$\left[u_{0j}\right] \sim \text{N}(0,\ \Omega_u)\ :\ \Omega_u = \left[0.037(0.009)\right]$

$\text{var}(\text{obs}_{ij}|\pi_{ij}) = \pi_{ij}$

Deviance(MCMC) = 2026.424(354 of 354 cases in use)

and if we investigate for example the chain for the Belgium effect, β_1, (via the **Trajectories** window) we see the following:



Accuracy Diagnostics
Raftery-Lewis (quantile) : Nhat = (159431,281815)
when q = (0.025,0.975), r = 0.005 and s = 0.95
Brooks-Draper (mean) : Nhat = 76991512
when k = 2 sigfigs and alpha = 0.05

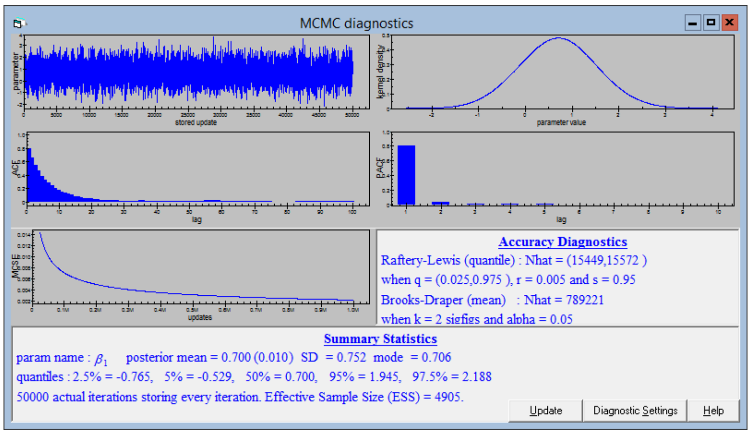
Summary Statistics
param name : β_1 posterior mean = 0.762 (0.100) SD = 0.676 mode = 0.579
quantiles : 2.5% = -0.437, 5% = -0.285, 50% = 0.715, 95% = 1.926, 97.5% = 2.229
50000 actual iterations storing every iteration. Effective Sample Size (ESS) = 40.

Here we have an ESS of 40 after in reality running for 50,000 iterations. We will again try and improve matters by using an orthogonal parameterisation as follows:

- Click on the **IGLS/RIGLS** tab on the **Estimation control** window.
- Click **Start**.
- Click on the **Estimation Control** button
- Select the tab labelled **MCMC** on the **Estimation control** window.

- Select **MCMC/MCMC options** from the **Model** menu.
- Click on **Use orthogonal parameterisation**
- Click on the **Done** button on the **MCMC options** window.
- Click **Start**.

If we now look at the β_1 diagnostics we see the following:



Accuracy Diagnostics
Raftery-Lewis (quantile) : Nhat = (15449,15572)
when q = (0.025,0.975), r = 0.005 and s = 0.95
Brooks-Draper (mean) : Nhat = 789221
when k = 2 sigfigs and alpha = 0.05

Summary Statistics
param name : β_1 posterior mean = 0.700 (0.010) SD = 0.752 mode = 0.706
quantiles : 2.5% = -0.765, 5% = -0.529, 50% = 0.700, 95% = 1.945, 97.5% = 2.188
50000 actual iterations storing every iteration. Effective Sample Size (ESS) = 4905.

Here the effective sample size increases from 40 to 4905, a factor of over a 100! If we look at the table below we see universal improvement with the orthognal parameterisation.

Parameter	Non-orthogonal	Orthogonal
β_1	40	4905
β_2	178	650
β_3	38	5315
β_4	1697	1764
β_5	73	1005
β_6	410	1640
β_7	88	6770
β_8	27	2220
β_9	29	2360
β_{10}	40	5104
β_{11}	211	1052
β_{12}	38	5307
β_{13}	1769	1964
β_{14}	77	1106
β_{15}	464	1764
β_{16}	87	6643
β_{17}	27	2190
β_{18}	29	2352
σ^2_{u0}	4132	4248

The improvement varies from less than twice as big an ESS to over 150 times as big however the important statistic is probably the value of the smallest ESS and this rises from 27 to 650 which is a great improvement. For this model at least we can actually interpret the orthogonal parameterisation. The first 9 predictors are already orthogonal and so will remain the same in the orthogonal parameterisation. The other nine (interaction) terms are orthogonal to all predictors apart from the corresponding intercepts and so to make them orthogonal they will be centred around the nation mean for UVB. This also explains in part the great differences between the ESS values in the original parameterisation as each nation has a different mean UVB score and for example France (nation 4) has the closest to 0.

We will next show how orthogonalisation helps with an ordered multinomial model.

23.5 An Ordered multinomial example

In Chapter 13 we looked at fitting ordered response multinomial models to a dataset of A level (chemistry) exam results. These responses are grades from A - F on an exam taken at age 18 in the UK. We will here look at the effect of using orthogonal fixed effects on the multilevel model described in section 20.5. Once again we need to load up the worksheet and set up the model before we can compare methods. This can be done as follows:

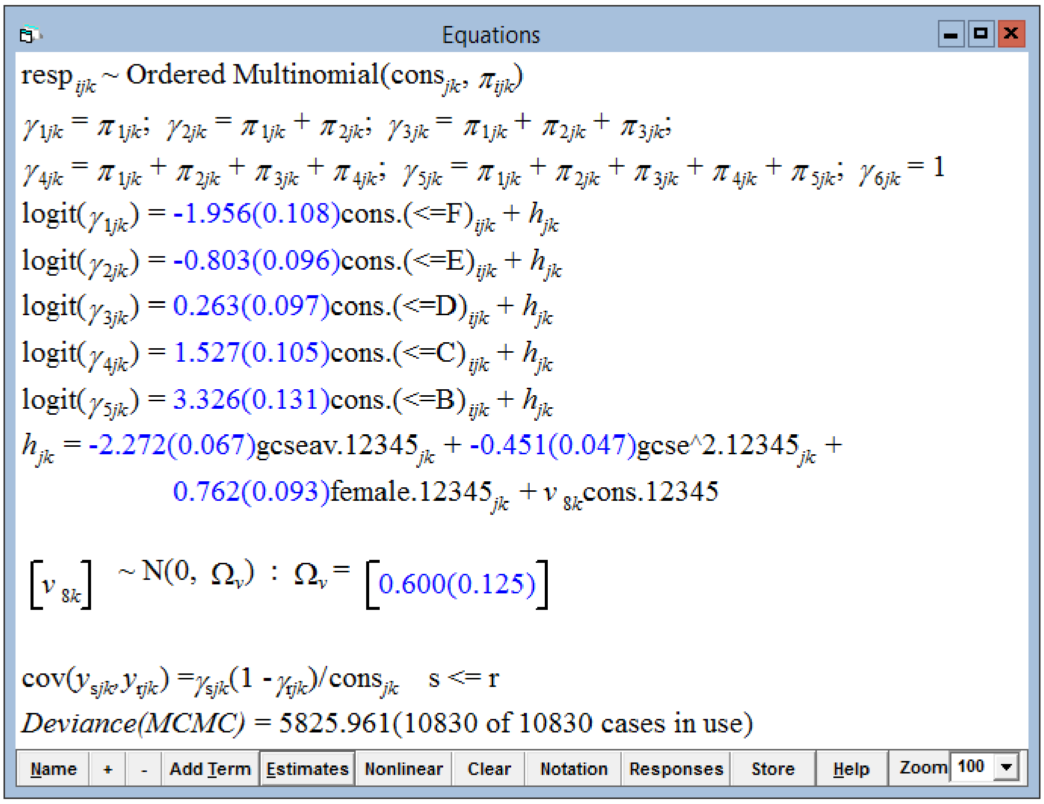
- Select **Open sample worksheet** from the **File** menu.
- Select **alevchem.ws** from the list of worksheets.
- Select **Open**.
- Select the **Command interface** option from the **Data Manipulation** menu.
- Type the following commands into the **Command interface** window.

```
▶ CALC c9=c5/c6
▶ CALC c9=c9-6
▶ CALC c10=c9^2
▶ NAME c9 'gcseav' c10 'gcse^2'
```

- Select **Equations** from the **Model** menu.
- Click on the red *y* and again select **a-point** as the *y* variable.
- Select **2-ij** as the number of models, **estab** as **level 2(j)** and **pupil** as **level 1(i)** identifier.
- Click on the **Done** button.

- Click on the **N** and from the list of distributions that appears scroll down and select **multinomial**.
- In the **Multinomial options** box select **Ordered proportional odds** and **A** as the **reference category**.
- Click on the **Done** button.
- Click on the n_{jk} in the window and select **cons** as the **denominator**.
- Click on the **Done** button.
- Click on the **Add Term** button and select **cons** from the **variable** list.
- Click on the **add Separate coefficients** button.
- Click on the **Add Term** button.
- Select **gcseav** from the **variable** list.
- Click on the **add Common coefficient** button.
- Click on the **Include all** and **Done** buttons.
- Repeat this procedure for the variables **gcse^2**, and **gender**.
- Click on the **Add Term** button.
- Select **cons** from the **variable** list.
- Click on the **add Common coefficient** button.
- Click on the **Include all** and **Done** buttons.
- Click on the **cons.12345** term.
- In the **X variable** window that appears tick the **k(estab_long)** tickbox.
- Remove the tick in the **Fixed Parameter** tickbox.
- Click on the **Done** button.
- Click on the **Start** button.
- Change **Estimation method** to **MCMC**.
- Click on the **Start** button.

Upon completion of the iterations we get the following estimates:



$$\text{resp}_{ijk} \sim \text{Ordered Multinomial}(\text{cons}_{jk}, \pi_{ijk})$$

$$\gamma_{1jk} = \pi_{1jk};\ \gamma_{2jk} = \pi_{1jk} + \pi_{2jk};\ \gamma_{3jk} = \pi_{1jk} + \pi_{2jk} + \pi_{3jk};$$

$$\gamma_{4jk} = \pi_{1jk} + \pi_{2jk} + \pi_{3jk} + \pi_{4jk};\ \gamma_{5jk} = \pi_{1jk} + \pi_{2jk} + \pi_{3jk} + \pi_{4jk} + \pi_{5jk};\ \gamma_{6jk} = 1$$

$$\text{logit}(\gamma_{1jk}) = -1.956(0.108)\text{cons.}(\text{<=F})_{ijk} + h_{jk}$$

$$\text{logit}(\gamma_{2jk}) = -0.803(0.096)\text{cons.}(\text{<=E})_{ijk} + h_{jk}$$

$$\text{logit}(\gamma_{3jk}) = 0.263(0.097)\text{cons.}(\text{<=D})_{ijk} + h_{jk}$$

$$\text{logit}(\gamma_{4jk}) = 1.527(0.105)\text{cons.}(\text{<=C})_{ijk} + h_{jk}$$

$$\text{logit}(\gamma_{5jk}) = 3.326(0.131)\text{cons.}(\text{<=B})_{ijk} + h_{jk}$$

$$h_{jk} = -2.272(0.067)\text{gcseav.12345}_{jk} + -0.451(0.047)\text{gcse\^{}2.12345}_{jk} + 0.762(0.093)\text{female.12345}_{jk} + v_{8k}\text{cons.12345}$$

$$\begin{bmatrix} v_{8k} \end{bmatrix} \sim \text{N}(0,\ \Omega_v)\ :\ \Omega_v = \begin{bmatrix} 0.600(0.125) \end{bmatrix}$$

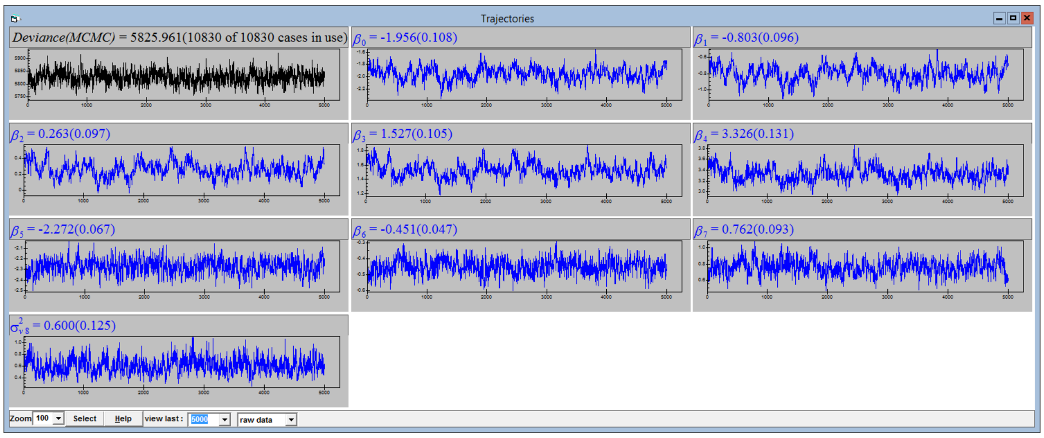
$$\text{cov}(y_{sjk}, y_{rjk}) = \gamma_{sjk}(1 - \gamma_{rjk})/\text{cons}_{jk} \quad s <= r$$

Deviance(MCMC) = 5825.961(10830 of 10830 cases in use)

If we take a look at the **Trajectories** by doing the following:

- Select **Trajectories** from the **Model** menu.
- Click on **Select** on the **Trajectories** and on the window that appears choose **3 graphs per row**.
- Click on the **Done** button.
- Change **view last** to **5000**.

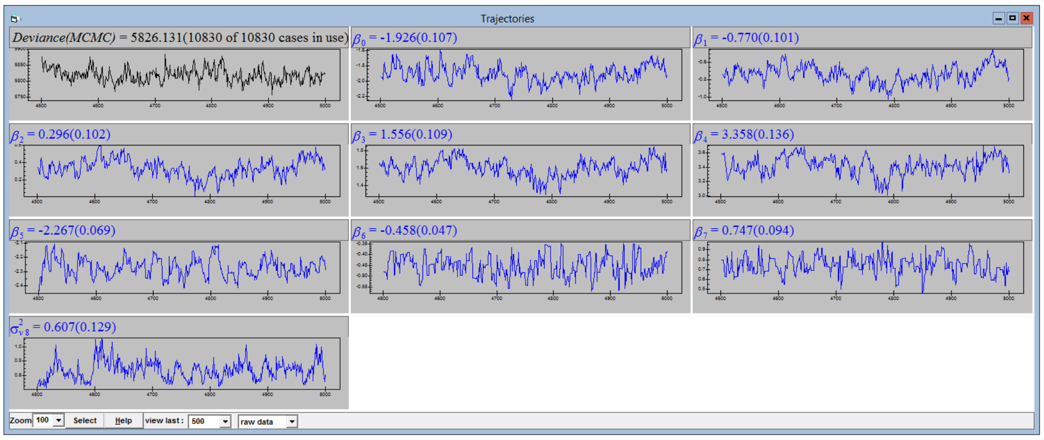
we see that the chains' mixing is not too bad:



We will next use the orthogonal predictor option:

- Click on the **IGLS/RIGLS** tab on the **Estimation control** window.
- Click **Start**.
- Click on the **Estimation Control** button
- Select the tab labelled **MCMC** on the **Estimation control** window.
- Select **MCMC/MCMC options** from the **Model** menu.
- Click on **Use orthogonal parameterisation**
- Click on the **Done** button on the **MCMC options** window.
- Click **Start**.

Having run the model using the orthogonal parameterisation we can look again at the **Trajectories** window and see that things have improved although not by as huge an amount as we observed with the Poisson example.


Trajectories
Deviance(MCMC) = 5826.131(10830 of 10830 cases in use)
β_0 = -1.926(0.107)
β_1 = -0.770(0.101)
β_2 = 0.296(0.102)
β_3 = 1.556(0.109)
β_4 = 3.358(0.136)
β_5 = -2.267(0.069)
β_6 = -0.458(0.047)
β_7 = 0.747(0.094)
σ^2_{v8} = 0.607(0.129)
Zoom 100
Select
Help
view last : 500
raw data

We can look closely at each parameter chain and pick out the ESS and these are given in the following table:

Parameter	Non-orthogonal	Orthogonal
β_0	102	227
β_1	80	158
β_2	72	129
β_3	67	124
β_4	71	155
β_5	431	282
β_6	191	754
β_7	209	795
σ^2_{v8}	239	193

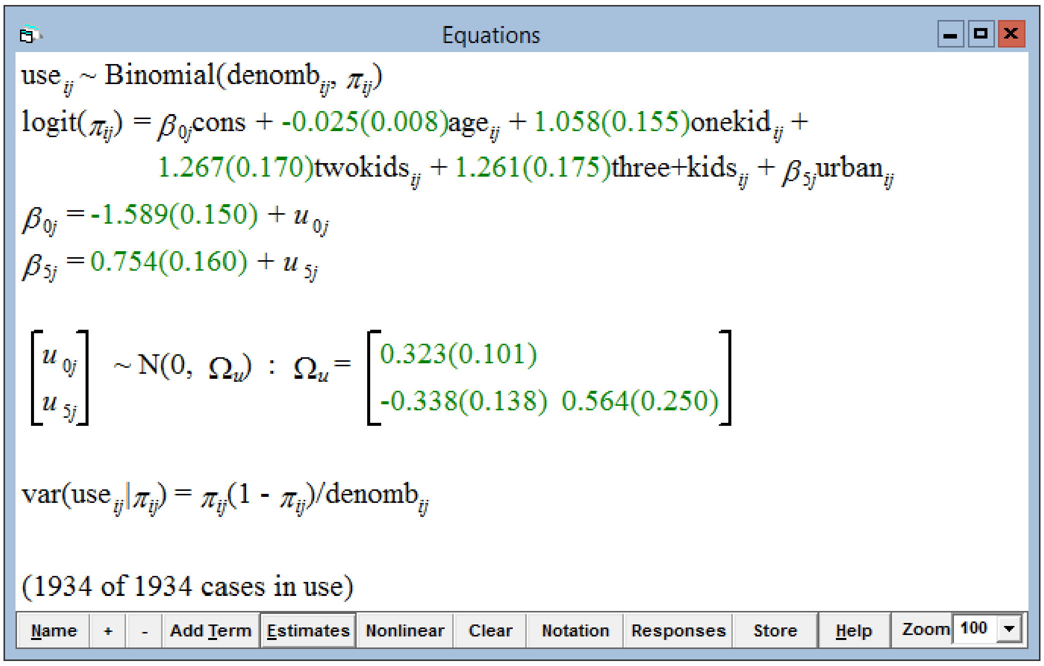
Here we see that in general the method has improved the mixing (with the exception of β_5); the improvement is of the order of a factor of around 2 in terms of increased sample size which is far less than for the Poisson models. Note, although we do not give an example of it here, it is also possible to use orthogonal parameterisations with unordered multinomial examples.

23.6 The WinBUGS interface

The MLwiN to WinBUGS interface produces for the user WinBUGS code for the same model that they are planning to fit in MLwiN. As we saw in Chapter 7 this then allows the user to modify their code further to use features unavailable in MLwiN.

The new MCMC methods that are described in these chapters will also translate (in part) via the WinBUGS interface. The previous two chapter methods (SMCMC and SMVN) are not available in WinBUGS and so are not included; however orthogonal parameterisations are as we describe here for the Bangladeshi example.

To begin this section we need to once again set up the Bangladeshi model as detailed in section 23.3. If you follow the instructions for loading the worksheet and setting up the model and then run just with IGLS the **Equations** window should then look as follows:



$\text{use}_{ij} \sim \text{Binomial}(\text{denomb}_{ij}, \pi_{ij})$

$\text{logit}(\pi_{ij}) = \beta_{0j}\text{cons} + -0.025(0.008)\text{age}_{ij} + 1.058(0.155)\text{onekid}_{ij} + 1.267(0.170)\text{twokids}_{ij} + 1.261(0.175)\text{three+kids}_{ij} + \beta_{5j}\text{urban}_{ij}$

$\beta_{0j} = -1.589(0.150) + u_{0j}$

$\beta_{5j} = 0.754(0.160) + u_{5j}$

$$\begin{bmatrix} u_{0j} \\ u_{5j} \end{bmatrix} \sim \text{N}(0,\ \Omega_u)\ :\ \Omega_u = \begin{bmatrix} 0.323(0.101) & \\ -0.338(0.138) & 0.564(0.250) \end{bmatrix}$$

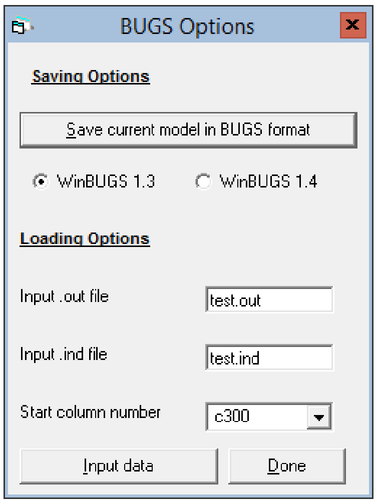
$\text{var}(\text{use}_{ij}|\pi_{ij}) = \pi_{ij}(1 - \pi_{ij})/\text{denomb}_{ij}$

(1934 of 1934 cases in use)

In Chapter 7 we described WinBUGS in some details and will assume the reader has already read this chapter. We will now need to ensure we have orthogonal predictors set up and then get to the BUGS options in MLwiN: we need to do the following:

- Click on the **Estimation Control** button.
- Select the **MCMC** tab.
- Select **MCMC/MCMC options** from the **Model** menu.
- Click on **Use orthogonal parameterisation**.
- Click on the **Done** button on the **MCMC options** window.
- Select **MCMC/WinBUGS Options** from the **Model** menu.

This will bring up the **WinBUGS Options** screen that looks as follows:



WinBUGS 1.3 [✓] WinBUGS 1.4

Input .out file: test.out

Input .ind file: test.ind

Start column number: c300

From this screen we can save the BUGS code for the currently set up model or read in the output files that contain parameter traces from BUGS for use in MLwiN (see later). For now we will save our current model in BUGS format:

- Select the **WinBUGS 1.4** button.
- Click on the large button at the top of the window.

This will bring up a file save window similar to those for inputting and saving worksheets. For now we will save the file in the default directory as **bangort.bug**. This will create a file that contains the BUGS model definition, initial values and data.

To now fit our model in WinBUGS, we must start the WinBUGS program and read in the file **bangort.bug** (from the directory it was saved in) as a text file. Note that you will have to change the **Files of type** box to **All files (*.*)** to see the file **bangort.bug**. Having read in the file, a window headed **bangort.bug** will appear containing the information needed by BUGS for this model.

We will here detail the model code created for WinBUGS in chunks. The first section detailed below specifies the distribution for the variable **use** in terms of a set of orthogonal predictors (**orthog**) with associated coefficients (**betaort**)

```
# WINBUGS 1.4 code generated from MLwiN program

#----MODEL Definition----------------

model
{
# Level 1 definition
for(i in 1:N) {
use[i] ~ dbin(p[i],denom[i])
logit(p[i]) <- betaort[1] * orthog1[i]
+ betaort[2] * orthog2[i]
+ betaort[3] * orthog3[i]
+ betaort[4] * orthog4[i]
+ betaort[5] * orthog5[i]
+ betaort[6] * orthog6[i]
+ u2[district[i],1] * cons[i]
+ u2[district[i],2] * urban[i]
}
```

In the next section of code we need to specify the orthogonal predictors in terms of the original predictors as follows:

```
for(i in 1:N)
{
orthog1[i] <- 1.000000 * cons[i]
+ 0 * age[i]
+ 0 * onekid[i]
+ 0 * twokids[i]
+ 0 * three_kids[i]
+ 0 * urban[i]
orthog2[i] <- -0.002048 * cons[i]
+ 1 * age[i]
+ 0 * onekid[i]
+ 0 * twokids[i]
+ 0 * three_kids[i]
+ 0 * urban[i]
orthog3[i] <- -0.183058 * cons[i]
+ 0.00883829 * age[i]
+ 1 * onekid[i]
+ 0 * twokids[i]
+ 0 * three_kids[i]
+ 0 * urban[i]
orthog4[i] <- -0.195383 * cons[i]
+ 0.00122514 * age[i]
+ 0.200184 * onekid[i]
+ 1 * twokids[i]
+ 0 * three_kids[i]
+ 0 * urban[i]
orthog5[i] <- -0.552304 * cons[i]
+ -0.0305852 * age[i]
+ 0.432448 * onekid[i]
+ 0.56088 * twokids[i]
+ 1 * three_kids[i]
+ 0 * urban[i]
orthog6[i] <- -0.337310 * cons[i]
+ -0.00216548 * age[i]
+ 0.00678972 * onekid[i]
+ 0.0708158 * twokids[i]
+ 0.089127 * three_kids[i]
+ 1 * urban[i]
}
```

Here we see how the algorithm proceeds by adding in an additional predictor into the construction of each orthogonal predictor in turn and so the first predictor simply is the intercept whilst the second is a linear function of the intercept and age and so on. Formulating the model in this way means that in the data section later the original data rather than the orthogonal predictors can be given. The next section of the code relates the coefficients for the original predictors to those for the orthogonal predictors:

```
beta[1] <- 1.000000 * betaort[1]
+ -0.0020481 * betaort[2]
+ -0.183058 * betaort[3]
+ -0.195383 * betaort[4]
+ -0.552304 * betaort[5]
+ -0.33731 * betaort[6]
beta[2] <- 0.000000 * betaort[1]
+ 1 * betaort[2]
+ 0.00883829 * betaort[3]
+ 0.00122514 * betaort[4]
+ -0.0305852 * betaort[5]
+ -0.00216548 * betaort[6]
beta[3] <- 0.000000 * betaort[1]
+ 0 * betaort[2]
+ 1 * betaort[3]
+ 0.200184 * betaort[4]
+ 0.432448 * betaort[5]
+ 0.00678972 * betaort[6]
beta[4] <- 0.000000 * betaort[1]
+ 0 * betaort[2]
+ 0 * betaort[3]
+ 1 * betaort[4]
+ 0.56088 * betaort[5]
+ 0.0708158 * betaort[6]
beta[5] <- 0.000000 * betaort[1]
+ 0 * betaort[2]
+ 0 * betaort[3]
+ 0 * betaort[4]
+ 1 * betaort[5]
+ 0.089127 * betaort[6]
beta[6] <- 0.000000 * betaort[1]
+ 0 * betaort[2]
+ 0 * betaort[3]
+ 0 * betaort[4]
+ 0 * betaort[5]
+ 1 * betaort[6]
```

Here we see that, as the calculation of the coefficients involves a matrix multiplication with the transpose of the matrix for the predictors, the number of non-zero terms involved in each coefficient reduces by one with each coefficient in turn. Finally we have some code to define the rest of the model.

```
# Higher level definitions
for (j in 1:n2) {
u2[j,1:2] ~ dmnorm(zero2[1:2],tau.u2[1:2,1:2])
}
# Priors for fixed effects
for (k in 1:6) { betaort[k] ~ dflat() }
# Priors for random terms
for (i in 1:2) {zero2[i] <- 0}
tau.u2[1:2,1:2] ~ dwish(R2[1:2, 1:2],2)
sigma2.u2[1:2,1:2] <- inverse(tau.u2[,])
}
```

This code specifies the prior distributions for the random effects, the fixed effects and the level 2 variance. There then follows, as is customary, code for the initial values and the data.

Before running a model in WinBUGS we first need to read in the particular elements of the model using the **Specification** window available from the **Model** menu. After selecting the window containing the model by clicking on it, clicking on the **check model** button should give the message 'model is syntactically correct' at the bottom of the screen. As we saw earlier, WinBUGS by default will use a block updating algorithm developed by Gamerman (Gamerman, 1997) for the fixed effects which will not be improved by changing the parameterisation. We will therefore force WinBUGS not to use this method. To do this we need to select **Blocking Options** from the **Options** menu and remove the tick that appears there before closing the window.

Next we need to load in the data for the model. Due to the fact that the data is generally the largest part of the file generated by MLwiN it is included after the initial values. The data section always begins as follows:

```
#----Data File----------------------------------
list(
```

To load the data into BUGS we need to highlight the **list** identifier at the start of the data list and click on the **load data** button in the **specification** window. If this is successful the message 'data loaded' will appear at the bottom of the screen. Next we have to combine the data and model definition by clicking on the **compile** button. Again if this operation is successful a message appears at the bottom of the screen, this time stating 'model compiled'. Finally as BUGS uses MCMC methods all unknown parameters will need starting values. These are included in the initial values part of the file that starts as follows:

```
#----Initial values file----------------------------

list
```

To use these values in WinBUGS we need to highlight the **list** identifier at the start of the initial values and click on the **load inits** button on the **specification** window. This will then give the final message 'initial values loaded; model initialized'.

Before we start we have to tell WinBUGS which parameters we wish to monitor. We will choose the same parameters as MLwiN uses. From the **Inference** menu select the **Samples** options and a window will appear that allows the user to specify which parameters to monitor. In this window we will firstly select the fixed effects by typing **beta** in the **node** box. Note that when a correctly typed parameter is input the **set** button will become enabled. We will also want to use a burn-in of some iterations. We will also modify the **beg value** from **1** to **501** to allow a burn-in of 500 iterations. After this press the **set** button and the parameter will be set for monitoring. We now need to repeat this procedure with the variance matrix **sigma2.u2**.

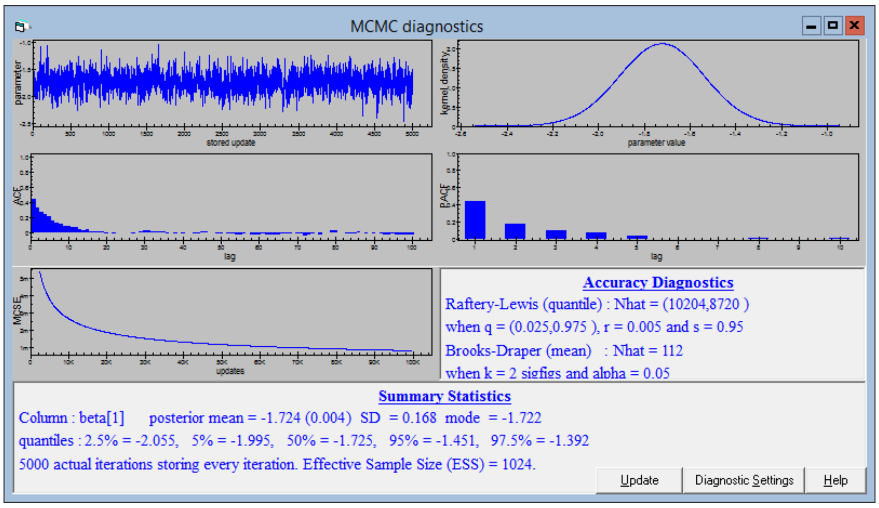
We are now ready to set the estimation engine running and this is done via the **Update** window found in the **Model** menu. We need to specify the number of updates (including the burn-in) and so we will replace the **1000** here with **5500** to give 5000 iterations after a burn-in of 500 iterations. We then press the **update** button to start the sampler. Note this updating will take some time in WinBUGS (235s on my machine).

As described in chapter 7 WinBUGS has some facilities to allow the user to get summary statistics and plots of the chains and the reader is welcome to investigate these here as well. For now however we will bring the WinBUGS chains back into MLwiN for further analysis. WinBUGS also has the option to produce input files in a format originally for a package called CODA. MLwiN can also use these files to input the parameter chains from WinBUGS into columns in MLwiN. Here we will consider all parameters by using the * option so select this in the **node** box and press the **coda** button on the **sample** window. This will produce two windows that are labelled **CODA index**, which contains the variable names, and **CODA for chain 1**, which contains the values for the parameter chains. We will now save these files as text files by clicking on the respective windows and then choosing **Save As** from the **File** menu. We will need to save the files in *plain text* (*.txt) format. We will store the **CODA index** file as **bangort.ind** and the **CODA for chain 1** file as **bangort.out** in the same directory as **bangort.bug**. Note that these are the extensions that the classic BUGS used for these files but, as we have selected the plain text format, WinBUGS will add an additional '.txt' to the first filename and so the files are actually saved as **bangort.ind.txt** and **bangort.out.txt**.

Now back in MLwiN if you want to input the traces return to the **BUGS options** window that we used earlier (available from the **Model** menu). Here we will need to modify the **.out** and **.in** file fields to **bangort.out** and **bangort.ind.txt** respectively. Note that if you did not put these files in the current directory you will have to include their full path names in the respective boxes. Pressing the **Input data** button will now load the chains into columns **c300** to **c309**. We can now use the MLwiN MCMC diagnostics on the BUGS output, for example for the intercept:

- Select the **Column Diagnostics** window from the **Basic Statistics** window.
- Select the column labelled **beta[1]**.
- Click on the **Apply** button.

Note that this parameter is the intercept that is labelled β_0 in MLwiN. You should now see the following diagnostics screen:



Accuracy Diagnostics
Raftery-Lewis (quantile) : Nhat = (10204,8720)
when q = (0.025,0.975), r = 0.005 and s = 0.95
Brooks-Draper (mean) : Nhat = 112
when k = 2 sigfigs and alpha = 0.05

Summary Statistics
Column : beta[1] posterior mean = -1.724 (0.004) SD = 0.168 mode = -1.722
quantiles : 2.5% = -2.055, 5% = -1.995, 50% = -1.725, 95% = -1.451, 97.5% = -1.392
5000 actual iterations storing every iteration. Effective Sample Size (ESS) = 1024.

Here we see a somewhat better ESS when compared with MLwiN. In the table below we give a list of estimates and ESS for both the MLwiN and WinBUGS software packages using the orthogonal parameterisation:

Parameter	MLwiN Estimate	MLwiN ESS	WinBUGS Estimate	WinBUGS ESS
β_0	-1.702 (0.167)	301	-1.724 (0.168)	1024
β_1	-0.026 (0.008)	1154	-0.027 (0.008)	4189
β_2	1.126 (0.162)	977	1.134 (0.161)	3718
β_3	1.349 (0.177)	1024	1.360 (0.176)	4224
β_4	1.346 (0.185)	924	1.360 (0.181)	3649
β_5	0.811 (0.192)	124	0.825 (0.179)	702
σ^2_{u0}	0.417 (0.135)	250	0.426 (0.138)	639
σ_{u50}	-0.419 (0.172)	167	-0.441 (0.187)	410
σ^2_{u5}	0.700 (0.300)	129	0.740 (0.338)	316

Here we see that WinBUGS gives somewhat better performance than MLwiN but it takes much longer: 235s as opposed to 25s. The orthogonalised code does run slower in WinBUGS than the standard code which takes 142s so the improvements, (not shown here) which consist of an ESS that has increase by a factor of between 2 and 6, have to be balanced with a slower run time. For this model one can also choose to run the default blocking option in WinBUGS. With this option in place and not orthogonalizing the predictors gives similar mixing and computation time (135s) to the standard code.

The WinBUGS program can be used with the other two examples here and we anticipate that there will be an advantage from using an orthogonal parameterisation for these as we have observed using MLwiN.

Chapter learning outcomes

- ⋆ How using orthogonal vectors improves mixing
- ⋆ The use of orthogonal vectors in binomial, Poisson and multinomial models
- ⋆ How to use orthogonal vectors and the WinBUGS interface

Chapter 24

Parameter expansion

In this chapter we introduce another reparameterisation method that can be used to improve the mixing of MCMC algorithms. One of the main causes of poor mixing in MCMC algorithms is correlation between model parameters. We have thus far seen two methods that remove the correlation between the fixed effects and residuals by (i) updating them together and (ii) integrating out the residuals from the model. We have then seen a reparameterisation method that accounts for correlations between fixed effects when they are updated individually. In this chapter we focus on another between parameters correlation, namely the correlation between a set of random effects and their variance. Generally the correlation between a set of random effects and their variance does not have a big impact on MCMC algorithms except in the situation when the variance has support near zero. In this case the algorithm can get stuck with chains close to zero for both the variance and the random effects for many iterations.

We will start by describing the method we will use to account for this problem before showing the method in action on both (i) a Normal response model with a large random effects variance and (ii) a binomial response model with a small random effects variance. We will also look at the effect of the prior for the variance, the WinBUGS interface and the extension of parameter expansion to random slopes models.

24.1 What is Parameter Expansion?

Parameter expansion is a method that was originally developed by Liu et al. (1998) to speed up the EM algorithm. This method was then considered in relation to the Gibbs sampler by Liu & Wu (1999) and has been considered particularly for random effect models by van Dyk & Meng (2001), Browne (2004) and Gelman et al. (2008). The method is called "parameter expansion" as the model we wish to fit is expanded by augmenting it with additional parameters to form an expanded model. These additional parameters make the model not identifiable but there exists within the model an 'embedded model' which is identifiable and is the model that we wish to fit. This means that the original parameters can be constructed from the new augmented parameter set.

Let us consider as a simple example a 2-level variance components model for the **tutorial** dataset. Here we have the exam marks of pupils clustered as they are taught in groups in a set of schools. Let y_{ij} be the mark for pupil i in school j, then let us assume we have one predictor variable, the (standardised) mark obtained in a reading test taken at age 11, x_{ij}. A two level variance components model can then be described as follows:

$$y_{ij} = \beta_0 + \beta_1 x_{ij} + u_j + e_{ij}, \quad u_j \sim \mathrm{N}(0, \sigma_u^2), \quad e_{ij} \sim \mathrm{N}(0, \sigma_e^2)$$

Here we have an average intercept β_0, and slope with reading test β_1; school level residuals u_j with variance σ_u^2; and pupil level residuals e_{ij} with variance σ_e^2.

For a Bayesian model we will here use 'diffuse' priors as follows:

$$p(\beta_0) \propto 1, \quad p(\beta_1) \propto 1, \quad p(1/\sigma_u^2) \sim \Gamma(\varepsilon, \varepsilon), \quad p(1/\sigma_e^2) \sim \Gamma(\varepsilon, \varepsilon)$$

In fact in the tutorial example we have significant large school effects and so the mixing of the school level variance parameter is good. We will however still illustrate the parameter expansion technique.

To reparameterize the above model we will introduce an additional parameter α as follows:

$$y_{ij} = \beta_0 + \beta_1 x_{ij} + \alpha u_j + e_{ij}, \quad u_j \sim \mathrm{N}(0, \sigma_u^2), \quad e_{ij} \sim \mathrm{N}(0, \sigma_e^2)$$

We give α a flat Uniform prior although other alternatives here might be Normal priors with large variances. As mentioned in Browne et al. (2009*b*) constraining the term σ_u^2 to the value 1 would see $|\alpha|$ playing the role of the standard deviation of the random effects and we would then effectively have a prior that was uniform on the standard deviation scale. This option is however not offered in MLwiN.

The MCMC algorithm is not greatly changed by the addition of α as the only existing update steps that change are the steps for the u_j. u_j will still have a Normal conditional posterior distribution but this will additionally depend on the value of α. There will also need to be a Gibbs sampling step for α which will also have a Normal conditional posterior distribution.

As we will see more clearly in the WinBUGS section at the end of the chapter the parameters α, u_j and σ_u^2 will not be identifiable and their chains will appear to wander fairly randomly. We can however form $u_j^* = \alpha u_j$ and $\sigma_{u*}^2 = \alpha^2\sigma_u^2$ which represent the original parameters and these will be identifiable parameters.

These two equations really sum up why parameter expansion improves mixing. Basically if there is support near zero for the level 2 variance then the chains can get stuck for long periods with both the variance and the residuals being close to zero. The chains can escape to explore the rest of the parameter space but when they return close to zero they will get stuck again as it is difficult to move one parameter without also moving the others away from zero and each parameter is updated separately. The parameter α circumvents this problem as an increase in α will in fact move both the variance and the residuals together.

It should be noted that the prior distribution for σ_{u*}^2 will not now be $\Gamma^{-1}(\varepsilon, \varepsilon)$ as we used in previous chapters as σ_u^2 has this prior. As discussed in Gelman (2006) the prior is one of the folded-noncentral-t distributions although for now we will simply think of it as an alternative non-informative prior. We will now consider using parameter expansion in MLwiN.

24.2 The tutorial example

We will first load up the **tutorial** worksheet in MLwiN and fit a variance components model as described above to the data. Firstly we need to retrieve the worksheet:

- Select **Open sample worksheet** from the **File** menu.
- Select **tutorial.ws**.

This will open the **Names** window and we now need to bring up the **Equations** window and set up the model.

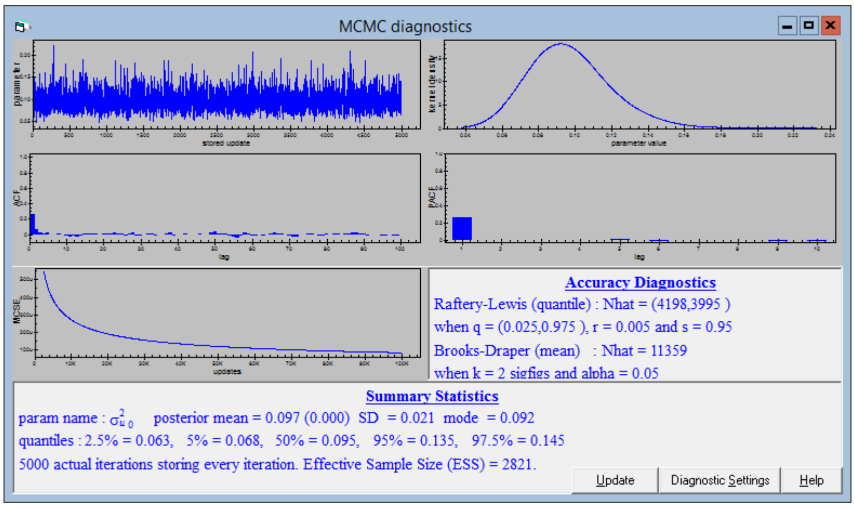
- Select **Equations** from the **Model** menu.
- Click on **y** (either of the **y** symbols shown will do).
- In the **y** list, select **normexam**.
- In the **N levels** list, select **2-ij**.
- In the **level 2(j):** list, select **school**.
- In the **level 1(i):** list, select **student**.
- Click on the **Done** button.

- Click on the red x_0.
- In the drop-down list, select **cons**.
- Check the box labelled **i(student)**.
- Check the box labelled **j(school)**.
- Click on the **Done** button.
- Click the **Add Term** button on the **Equations** window tool bar.
- Select **standlrt** from the **variable** list.
- Click on the **Done** button.

These commands will set up the model which we now need to fit using MCMC, although firstly as usual we will fit it using IGLS to get starting values.

- Click **Start**.
- Click on the **Estimation Control** button
- Select the tab labelled **MCMC**
- Click **Start**.
- Select **Trajectories** from the **Model** menu.

If we now look at the chain for the level 2 variance σ^2_{u0} we see the following:



Accuracy Diagnostics
Raftery-Lewis (quantile) : Nhat = (4198,3995)
when q = (0.025,0.975), r = 0.005 and s = 0.95
Brooks-Draper (mean) : Nhat = 11359
when k = 2 sigfigs and alpha = 0.05

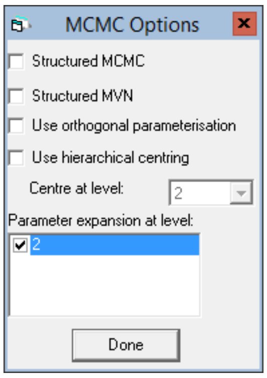
Summary Statistics
param name : σ^2_{u0} posterior mean = 0.097 (0.000) SD = 0.021 mode = 0.092
quantiles : 2.5% = 0.063, 5% = 0.068, 50% = 0.095, 95% = 0.135, 97.5% = 0.145
5000 actual iterations storing every iteration. Effective Sample Size (ESS) = 2821.

Here the posterior mean estimate is 0.097 with posterior mode 0.092 which is the same as the IGLS estimate. We also see an effective sample size of 2,821 which is reasonably large.

We will next use parameter expansion on this model.

- Click on the **IGLS/RIGLS** tab on the **Estimation control** window.
- Click **Start**.
- Click on the **Estimation Control** button.
- Select the tab labelled **MCMC** on the **Estimation control** window.
- Select **MCMC/MCMC options** from the **Model** menu.
- Click on the level **2** tick box under **Parameter expansion at level:**.

The **MCMC options** window should then look as follows:



☐ Structured MCMC

☐ Structured MVN

☐ Use orthogonal parameterisation

☐ Use hierarchical centring

Centre at level: 2

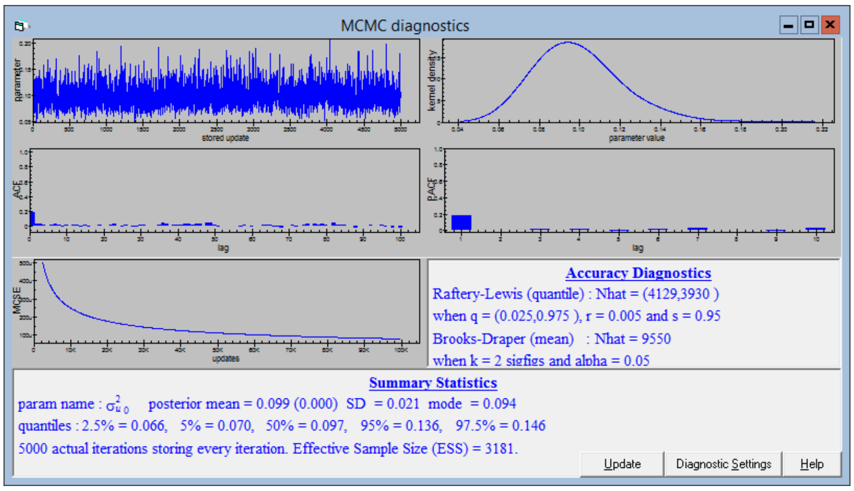
Parameter expansion at level:

☑ 2

We will now run the model using parameter expansion by doing the following:

- Click on the **Done** button on the **MCMC options** window.
- Click **Start**.

If we again look at the diagnostics for σ^2_{u0} (via the **Trajectories** window) we will see the following:



Accuracy Diagnostics

Raftery-Lewis (quantile) : Nhat = (4129,3930)
when q = (0.025,0.975), r = 0.005 and s = 0.95
Brooks-Draper (mean) : Nhat = 9550
when k = 2 sigfigs and alpha = 0.05

Summary Statistics

param name : σ^2_{u0} posterior mean = 0.099 (0.000) SD = 0.021 mode = 0.094
quantiles : 2.5% = 0.066, 5% = 0.070, 50% = 0.097, 95% = 0.136, 97.5% = 0.146
5000 actual iterations storing every iteration. Effective Sample Size (ESS) = 3181.

Here we see that the estimates have increased slightly with a posterior mean estimate of 0.099 and posterior mode 0.094. This is expected as the prior has changed and the new prior does not pull the variance towards zero as much as the $\Gamma^{-1}(\varepsilon, \varepsilon)$ prior. We also see a slightly larger effective sample size of 3,182. As the posterior has little mass near zero the benefits of parameter expansion are in this example fairly minimal.

What should be made clear here is that in MLwiN the **Equations** window is not changed to reflect the parameter expansion option and the transformation from the non-identified parameters to the identified parameters is done in the background. It does however show the correct estimates. We will next look at an example where parameter expansion has more of an impact.

24.3 Binary responses - Voting example

Historically the MLwiN documentation used as its binary response example a dataset on voting intentions in the UK. This example was replaced as the clustering in the data was quite small and hence the Bangladeshi dataset was used instead. An example with small amounts of clustering is however a good illustration for parameter expansion. Parameter expansion in a random effects logistic regression works in a very similar way to the Normal response model in the last section. Basically an α parameter is introduced into the model that multiplies the random effects in the linear predictor. This parameter then is involved in both the random effects and their variance and hence avoids the sticking near zero problem described previously. In terms of an MCMC algorithm, in MLwiN the u_j are updated via Metropolis sampling in logistic regression models and this will be true of the parameter expanded formulation also. There will also be an additional Metropolis step to update the α parameter.

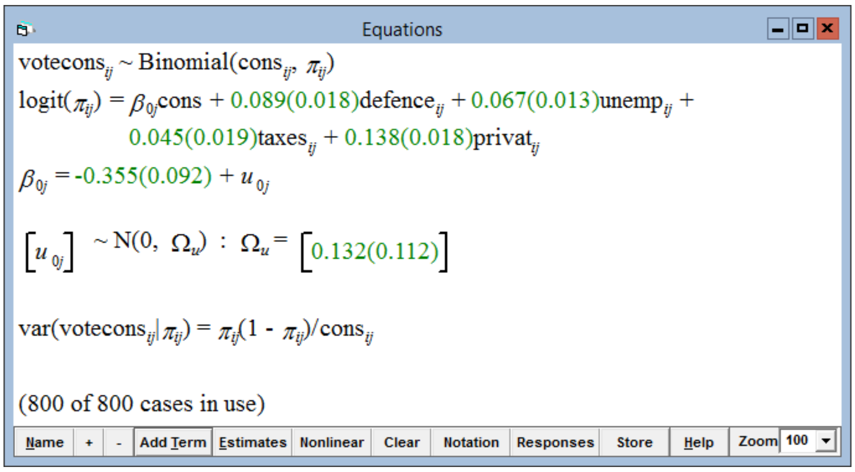
We will firstly describe briefly the background of the BES83 dataset before giving instructions on how to set up the model of interest. The data we will consider come from the longitudinal component of the British Election Study and consists of voting intentions in the 1983 UK general elections of 800 voters who were grouped within 110 constituencies . The response variable is whether or not the voter intended to vote Conservative (the party in power at the time). The two level structure of voters nested within constituencies suggested fitting a random effects logistic regression model.

There are also several interesting attitudinal predictors that relate the voters' opinions on four topical issues of the 1980s. The four such predictors were the voters' attitudes on nuclear weapons, unemployment, tax cuts and privatisation of public services. Each of these were measured on a 21-point scale designed so that higher scores were expected to be related to more right-wing views.

Firstly we need to retrieve the worksheet and set up the model:

- Select **Open sample worksheet** from the **File** menu.
- Select **bes83.ws**.
- Select the **Open** button.
- Select **Equations** from the **Model** menu.
- Click on the red y.
- Select **votecons** for the **y** variable.
- Select **2-ij** for the **number (N) of levels**.
- Select **area** as **level 2(j)**.
- Select **voter** as **level 1(i)**.
- Click on the **done** button.
- Click on the **N** and instead choose **Binomial** from the list.
- Click on the **Done** button.
- Click on the red n_{ij}.
- Select **cons** and click on the **Done** button.
- Click on the red x_0.
- Select **cons**.
- Click on the **(j)area** box in the **X variable** window.
- Click on the **Done** button. (Note the level 1 variance is Binomial and so doesn't need adding).
- Click on the **Add Term** button.
- Select **defence** from the **variable** list and click on the **Done** button.
- Click on the **Add Term** button on the **Equations** window.
- Select **unemp** from the **variable** pull-down list.
- Click on the **Done** button.
- Click on the **Add Term** button on the **Equations** window.
- Choose **taxes** from the **Variable** list.
- Click on the **Done** button.
- Click on the **Add Term** button on the **Equations** window.
- Choose **privat** from the **Variable** list.
- Click on the **Done** button.
- Click on the **Start** button.

The above will have set up the model and run it using IGLS (1st order MQL) estimation. The **Equations** window will then look as follows:


Equations
Name + - Add Term Estimates Nonlinear Clear Notation Responses Store Help Zoom 100

$\text{votecons}_{ij} \sim \text{Binomial}(\text{cons}_{ij}, \pi_{ij})$

$\text{logit}(\pi_{ij}) = \beta_{0j}\text{cons} + 0.089(0.018)\text{defence}_{ij} + 0.067(0.013)\text{unemp}_{ij} + 0.045(0.019)\text{taxes}_{ij} + 0.138(0.018)\text{privat}_{ij}$

$\beta_{0j} = -0.355(0.092) + u_{0j}$

$\left[u_{0j}\right] \sim \text{N}(0,\ \Omega_u)\ :\ \Omega_u = \left[0.132(0.112)\right]$

$\text{var}(\text{votecons}_{ij}|\pi_{ij}) = \pi_{ij}(1 - \pi_{ij})/\text{cons}_{ij}$

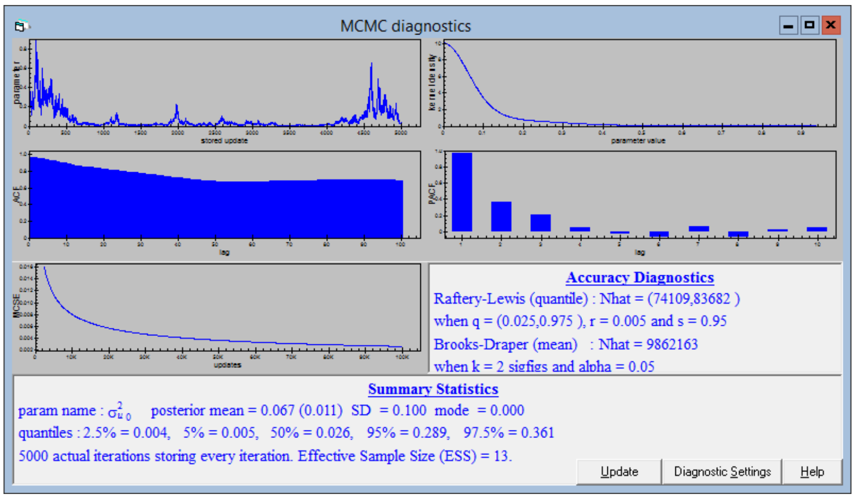
(800 of 800 cases in use)

Here we see positive coefficients for each of the 4 predictors as we were expecting given that a higher score suggests more right-wing views. We can also see that the level 2 variance is not huge with a standard error nearly as big as the point estimate.

We will next fit the model using the standard MCMC algorithm

- Change **Estimation Method** to **MCMC**.
- Click on the **Start** button.
- Select **Trajectories** from the **Model** menu.
- Select the chain for σ^2_{u0} from the **Trajectories** window.

If we look at the MCMC diagnostics window we see the following:



Accuracy Diagnostics
Raftery-Lewis (quantile) : Nhat = (74109,83682)
when q = (0.025,0.975), r = 0.005 and s = 0.95
Brooks-Draper (mean) : Nhat = 9862163
when k = 2 sigfigs and alpha = 0.05

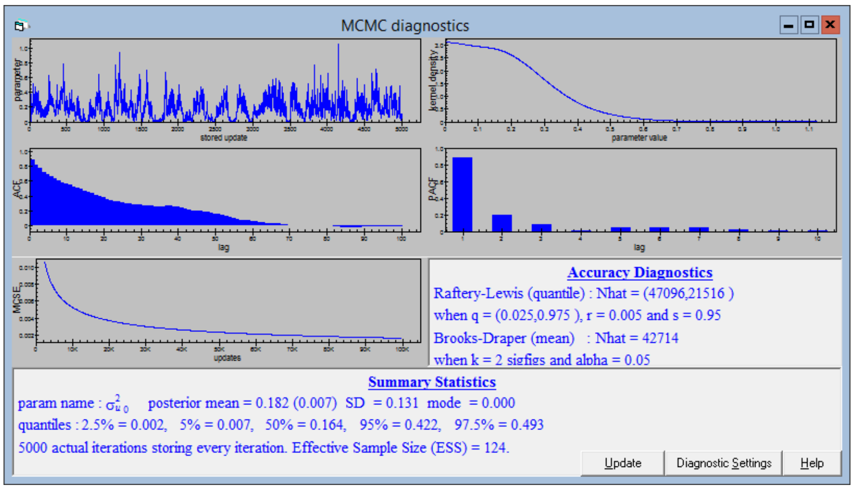
Summary Statistics
param name : σ^2_{u0} posterior mean = 0.067 (0.011) SD = 0.100 mode = 0.000
quantiles : 2.5% = 0.004, 5% = 0.005, 50% = 0.026, 95% = 0.289, 97.5% = 0.361
5000 actual iterations storing every iteration. Effective Sample Size (ESS) = 13.

Here we see that the mode of the kernel plot is 0 and there is not much evidence to in fact suggest a level 2 variance is needed. We also however see very poor mixing of the chain which results in an ESS of only 13! The estimate for σ^2_{u0} is also actually smaller than the estimate from IGLS.

We will next look at what happens when we choose to use parameter expansion with this example.

- Click on the **IGLS/RIGLS** tab on the **Estimation control** window.
- Click **Start**.
- Click on the **Estimation Control** button.
- Select the tab labelled **MCMC** on the **Estimation control** window.
- Select **MCMC/MCMC options** from the **Model** menu.
- Click on the **level 2 tick box** under **Parameter expansion at level:**.
- Click on the **Done** button on the **MCMC options** window.
- Click **Start**.

After running the model using parameter expansion we can once again look at the diagnostics for σ^2_{u0} as shown below:



Accuracy Diagnostics
Raftery-Lewis (quantile) : Nhat = (47096,21516)
when q = (0.025,0.975), r = 0.005 and s = 0.95
Brooks-Draper (mean) : Nhat = 42714
when k = 2 sigfigs and alpha = 0.05

Summary Statistics
param name : σ^2_{u0} posterior mean = 0.182 (0.007) SD = 0.131 mode = 0.000
quantiles : 2.5% = 0.002, 5% = 0.007, 50% = 0.164, 95% = 0.422, 97.5% = 0.493
5000 actual iterations storing every iteration. Effective Sample Size (ESS) = 124.

It is immediately clear here that this chain is mixing better and the ESS has increased to 124 from 13. The other thing to note is the dramatic change in posterior mean estimate. The variance was estimated as 0.132 by IGLS, 0.067 by MCMC originally and now is estimated as 0.182! It should of course be noted that the estimates are based on low ESS and so in the particular the original MCMC estimate could change a lot if the method was run for longer. If we run both parameterisations for 50,000 iterations we see an estimate of 0.123 for the original parameterisation with an ESS of 85 and an estimate of 0.195 for the parameter expanded model with an ESS of 855. What is more telling is that both parameterisations give a mode of 0. This explains in part one of the main issues with parameter expansion. Parameter expansion is clearly beneficial in models with small cluster variances but there remains the question of whether in such situations we should simply ignore the clustering.

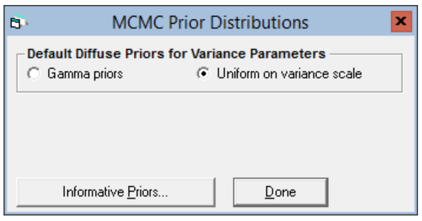
Another issue here is the question of whether the poor mixing is in fact simply due to the choice of prior distribution (given parameter expansion changes this). We will investigate this further in the next section.

24.4 The choice of prior distribution

MLwiN offers several choices of priors for the model parameters and in particular for variance parameters two sets of 'diffuse' priors are offered. The default is the inverse gamma priors but a uniform prior for σ^2_{u0} is also offered. To change to these priors we need to do the following:

- Click on the **IGLS/RIGLS** tab on the **Estimation control** window.
- Click **Start**.
- Click on the **Estimation Control** button.
- Select the tab labelled **MCMC** on the **Estimation control** window.
- Select **MCMC/MCMC options** from the **Model** menu.
- Remove the tick in the **level 2 tick box** under **Parameter expansion at level:**.
- Click on the **Done** button on the **MCMC options** window.
- Select **MCMC/Priors** from the **Model** menu.
- Click on the **Uniform on the Variance Scale** button.

The **Priors** window should then look as follows:

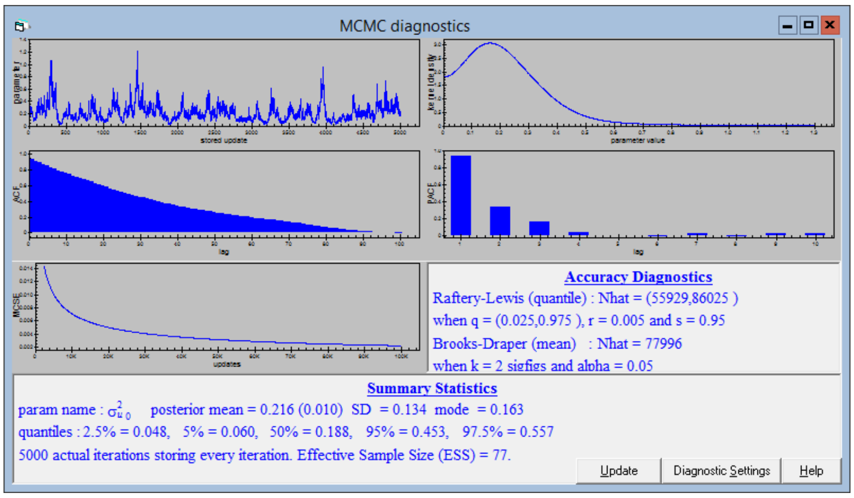

MCMC Prior Distributions

Default Diffuse Priors for Variance Parameters

Gamma priors [✓ Uniform on variance scale]

- Click on the **Done** button on the **Priors** window.
- Click **Start**.

MLwiN will now run the model with a uniform prior for σ^2_{u0} which tends to offer more support for larger values. The results of using this prior can be seen in the **Diagnostics** window as follows:



Accuracy Diagnostics
Raftery-Lewis (quantile) : Nhat = (55929,86025)
when q = (0.025,0.975), r = 0.005 and s = 0.95
Brooks-Draper (mean) : Nhat = 77996
when k = 2 sigfigs and alpha = 0.05

Summary Statistics
param name : σ^2_{u0} posterior mean = 0.216 (0.010) SD = 0.134 mode = 0.163
quantiles : 2.5% = 0.048, 5% = 0.060, 50% = 0.188, 95% = 0.453, 97.5% = 0.557
5000 actual iterations storing every iteration. Effective Sample Size (ESS) = 77.

Here we see that the Uniform prior does indeed increase the posterior mean estimate, here to 0.216. The mixing is also improved by the fact that the mode is no longer zero and the ESS is 77. This is better than with the inverse Gamma prior but not as good as with parameter expansion suggesting that the use of the parameter expansion algorithm has more benefit than simply changing the prior distribution would have to the mixing of the chains. If we run with this prior for 50,000 iterations the estimate increases to 0.255 with an ESS of 383.

We will not attempt to influence the reader here on which is the 'best' prior to use for the variance parameters in these models but instead refer them to Browne & Draper (2006) and Gelman (2006) for more on this subject.

24.5 Parameter expansion and WinBUGS

As we stated earlier MLwiN does not give the user any information via its **Equations** window as to whether parameter expansion is being used or not. We can however also use the interface to WinBUGS to create code that uses parameter expansion. For this we need to change our model so that we return to using inverse-Gamma priors and parameter expansion.

- Click on the **IGLS/RIGLS** tab on the **Estimation control** window.
- Click **Start**.
- Click on the **Estimation Control** button.
- Select the tab labelled **MCMC** on the **Estimation control** window.
- Select **MCMC/MCMC options** from the **Model** menu.
- Check the level **2** tick box under **Parameter expansion at level:**.
- Click on the **Done** button on the **MCMC options** window.

- Select **MCMC/Priors** from the **Model** menu.
- Click on the **Gamma priors** button.
- Click on the **Done** button.
- Select **MCMC/WinBUGS options** from the **Model** menu.
- Click on the **WinBUGS 1.4** button.
- Click on the big button at the top of the **WinBUGS options** window.
- Save the file as **votecons.bug** in the window that appears

We will then start up WinBUGS and look at the code that appears. We will need to read in the file **votecons.bug** (from the directory it was saved in) as a text file. To do this we will have to change the Files of type box to **All files (*.*)** to see the file **votecons.bug**. Having read in the file, a window headed **votecons.bug** will appear containing the information needed by BUGS for this model.

The model code for this model is fairly short and is detailed below:

```
# WINBUGS 1.4 code generated from MLwiN program


#----MODEL Definition----------------


model
{
# Level 1 definition
for(i in 1:N) {
VOTECONS[i] ~ dbin(p[i],denom[i])
logit(p[i]) <- beta[1] * cons[i]
+ beta[2] * defence[i]
+ beta[3] * unemp[i]
+ beta[4] * taxes[i]
+ beta[5] * privat[i]
+ alpha2 * u2[ares[i]] * cons[i]
}
# Higher level definitions
for (j in 1:n2) {
u2[j] ~ dnorm(0,tau.u2)
v2[j] <- u2[j]*alpha2
}
# Priors for fixed effects
for (k in 1:5) { beta[k] ~ dflat() }
alpha2 ~ dflat()
# Priors for random terms
tau.u2 ~ dgamma(0.001000,0.001000)
sigma2.u2 <- 1/tau.u2
sigma2.v2 <- sigma2.u2*alpha2*alpha2
}
```

Here we see initially the Binomial distribution for **VOTECONS** described and the linear predictor given for **p[i]**. Note here the parameter **alpha2** plays the role of α described in the earlier text. Basically with further levels/ classifications in a model we may have parameter expansion at further levels in which case we would have several α parameters and so the 2 indexes the classification. The WinBUGS code then gives prior distributions for **u2**, **beta**, **alpha2** and **tau.u2**. The lines to recover the original parameters which in WinBUGS are named **v2** and **sigma2.v2** are also given.

We will run this model in WinBUGS and then read the output files into MLwiN. Before running a model in WinBUGS we first need to read in the particular elements of the model using the **Specification** window available from the **Model** menu. After selecting the window containing the model by clicking on it, clicking on the **check model** button should give the message 'model is syntactically correct' at the bottom of the screen. As we have seen earlier we have options as to which MCMC methods WinBUGS uses for logistic regression models via the **blocking options** window available from the **Options** menu. As WinBUGS remembers the last used state of this option it is worth checking that the tick is present for fixed effects in this window to get the same estimates as are shown here.

Next we need to load in the data for the model. Due to the fact that the data is generally the largest part of the file generated by MLwiN it is included after the initial values. The data section always begins as follows:

```
#----Data File--------------------------------
list(
```

To load the data into BUGS we need to highlight the **list** identifier at the start of the data list and click on the **load data** button in the **specification** window. If this is successful the message 'data loaded' will appear at the bottom of the screen. Next we have to combine the data and model definition by clicking on the **compile** button. Again if this operation is successful a message appears at the bottom of the screen, this time stating 'model compiled'. Finally as BUGS uses MCMC methods all unknown parameters will need starting values. These are included in the initial values part of the file that starts as follows:

```
#----Initial values file---------------------------

list
```

To use these values in WinBUGS we need to highlight the **list** identifier at the start of the initial values and click on the **load inits** button on the **specification** window. This will then give the final message 'model is initialized'.

Before we start we have to tell WinBUGS which parameters we wish to monitor. We will here choose several parameters. From the **Inference** menu select the **Samples** options and a window will appear that allows the user to specify which parameters to monitor. In this window we will firstly select the fixed effects by typing **beta** in the **node** box. Note that when a correctly typed parameter is input the **set** button will become enabled. We will also want to use a burn-in of some iterations. We will modify the **beg value** from **1** to **501** to use a burn-in of 500. After this press the **set** button and the parameter will be set for monitoring. We now need to repeat this procedure with the level 2 variance **sigma2.v2**. We will also for illustration repeat the procedure with the unidentified parameters **sigma2.u2** and **alpha2**.

We are now ready to set the estimation engine running and this is done via the **Update** window found in the **Model** menu. We need to specify the number of updates (including the burn-in) and so we will replace the **1000** here with **5500** to give 5000 iterations after the burn-in as is used in MLwiN. We then press the **update** button to start the sampler.

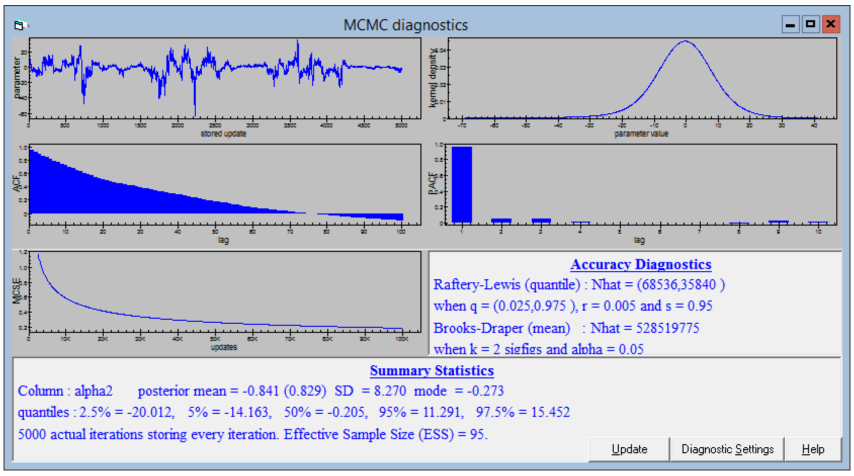
After running the model in WinBUGS we will bring the chains back into MLwiN for further analysis. WinBUGS also has the option to produce input files in a format originally for a package called CODA. MLwiN can also use these files to input the parameter chains from WinBUGS into columns in MLwiN. Here we will consider all parameters stored by using the ***** option so select this in the **node** box and press the **coda** button on the **sample** window. This will produce two windows that are labelled **CODA index** which contains the variable names and **CODA for chain 1** which contains the values for the parameter chains. We will now save these files as text files by clicking on the respective windows and then choosing **Save As** from the **File** menu. We will need to save the files in *plain text* (*.txt) format. We will store the **CODA index** file as **votecons.ind** and the **CODA for chain 1** file as **votecons.out** in the same directory as **votecons.bug**. Note that these are the extensions that the classic BUGS used for these files but, as we have selected the plain text format, WinBUGS will add an additional '.txt' to the first filename and so the files are actually saved as **votecons.ind.txt** and **votecons.out.txt**.

Now back in MLwiN if you want to input the traces return to the **BUGS options** window that we used earlier (available from the **Model** menu). Here we will need to modify the **.out** and **.in** file fields to **votecons.out.txt** and **votecons.ind.txt** respectively. Note that if you did not put these files in the current directory you will have to include their full path names in the respective boxes. Pressing the **Input data** button will now load the chains into columns **c300** to **c307**.

We can now use the MLwiN MCMC diagnostics on the BUGS output, for example if we want to look at the α_2 chain:

- Select the **Column Diagnostics** window from the **Basic Statistics** window.
- Select the column labelled **alpha2**.
- Click on the **Apply** button.

The following diagnostics will then appear:



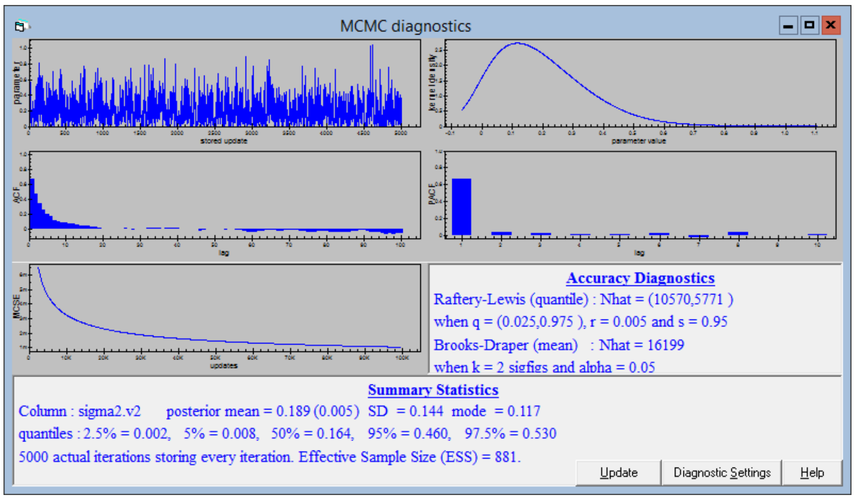
Accuracy Diagnostics
Raftery-Lewis (quantile) : Nhat = (68536,35840)
when q = (0.025,0.975), r = 0.005 and s = 0.95
Brooks-Draper (mean) : Nhat = 528519775
when k = 2 sigfigs and alpha = 0.05

Summary Statistics
Column : alpha2 posterior mean = -0.841 (0.829) SD = 8.270 mode = -0.273
quantiles : 2.5% = -20.012, 5% = -14.163, 50% = -0.205, 95% = 11.291, 97.5% = 15.452
5000 actual iterations storing every iteration. Effective Sample Size (ESS) = 95.

Here we can clearly see that this parameter is not identified as the chain just wanders rather aimlessly. The chain for **sigma2.u2** which is also not identified similarly wanders aimlessly; however if we look at the chain for **sigma2.v2**:

- Select the **Column Diagnostics** window from the **Basic Statistics** menu.
- Select the column labelled **sigma2.v2**.
- Click on the **Apply** button.

We see the following diagnostics which show a reasonably well behaved chain:



Accuracy Diagnostics
Raftery-Lewis (quantile) : Nhat = (10570,5771)
when q = (0.025,0.975), r = 0.005 and s = 0.95
Brooks-Draper (mean)   : Nhat = 16199
when k = 2 sigfigs and alpha = 0.05

Summary Statistics
Column : sigma2.v2   posterior mean = 0.189 (0.005) SD = 0.144 mode = 0.117
quantiles : 2.5% = 0.002, 5% = 0.008, 50% = 0.164, 95% = 0.460, 97.5% = 0.530
5000 actual iterations storing every iteration. Effective Sample Size (ESS) = 881.

Here the estimate of 0.189 is broadly similar to the estimate of 0.182 seen in MLwiN while the ESS of 881 is somewhat larger than the ESS of 124 . As we stated earlier not only do we have access to the chains for the non-identified parameters in WinBUGS but we can also change the model code, for example we could constrain sigma2.u2 to be 1 and then look at $|\alpha 2|$. This we will leave as an exercise for the reader.

24.6 Parameter expansion and random slopes

We finish this chapter by considering extensions of the parameter expansion method described to other models. Browne (2004) looked at the use of parameter expansion with cross-classified models and here each set of random effects could be expanded via the addition of an α parameter. Parameter expansion could also be used in conjunction with other reparameterisations, for example the orthogonal methods in the last chapter as illustrated in Browne et al. (2009*b*). Care has to be taken when considering the use of these methods with the hierarchical centring methods to be discussed in the next chapter and we would caution the reader not to use the methods together on the same classification, although as described in Browne (2004) they could be used together in the same model on different classifications.

The other extension that we can see for parameter expansion is to models with more than one set of random terms per classification, namely random slopes models. This case we believe has not been heavily studied although there are several possible ways that parameter expansion could be extended to random slopes models. We consider perhaps the simplest where we still have one α parameter that is introduced as a multiplier to all sets of random effects. Returning to the Normal model discussed earlier we would have

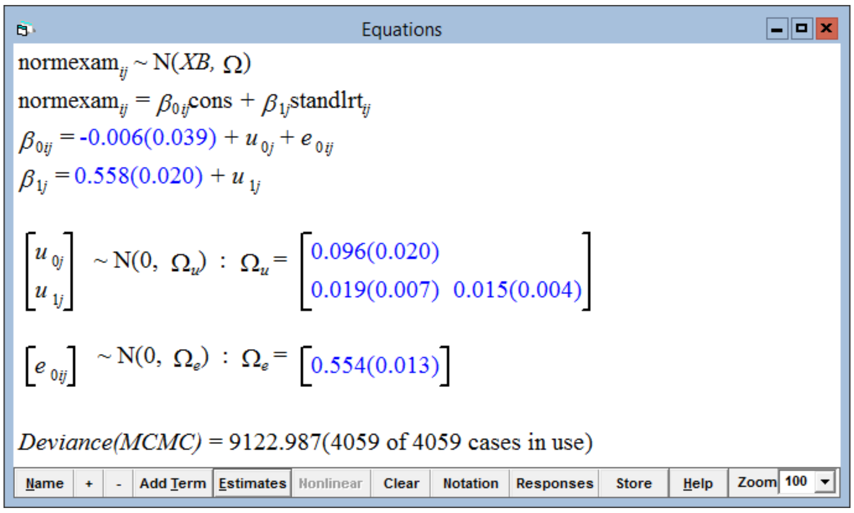
$$y_{ij} = \beta_0 + \beta_1 x_{ij} + \alpha u_{0j} + \alpha x_{ij} u_{1j} + e_{ij}, \quad u_j \sim \mathrm{N}(0, \Omega_u), \quad e_{ij} \sim \mathrm{N}(0, \sigma_e^2)$$

Here a random slope u_{1j} is introduced for each school and both the slopes and intercepts are multiplied by the same α in the linear predictor. Here to obtain the desired random effects and variance we simply have $u_{0j}^* = \alpha u_{0j}$, $u_{1j}^* = \alpha u_{1j}$ and $\Omega_u^* = \alpha^2 \Omega_u$. Typically in MLwiN we use a slightly informative Wishart prior for Ω_u so again we will have a change of prior for Ω_u^* but this will still be a slightly informative prior.

To demonstrate this in practice we need to fit the model in MLwiN. Rather than repeating the instructions at the start of the chapter here, the reader should follow the instructions given earlier to load up the **tutorial** worksheet and set up (but not estimate) the variance components model. When this model has been set up to modify it to the random slopes you will need to do the following:

- Click on the **standlrt** predictor.
- From the **X variable** window click in the **j(school)** tickbox.
- Click on the **Done** button.
- Click on the **Start** button.
- Change **Estimation Method** to MCMC.
- Click on the **Start** button.

This will run the model using the standard Gibbs sampling algorithm and produces the following estimates in the **Equations** window:



$\text{normexam}_{ij} \sim \text{N}(XB, \Omega)$

$\text{normexam}_{ij} = \beta_{0ij}\text{cons} + \beta_{1j}\text{standlrt}_{ij}$

$\beta_{0ij} = -0.006(0.039) + u_{0j} + e_{0ij}$

$\beta_{1j} = 0.558(0.020) + u_{1j}$

$$\begin{bmatrix} u_{0j} \\ u_{1j} \end{bmatrix} \sim \text{N}(0,\ \Omega_u) \ : \ \Omega_u = \begin{bmatrix} 0.096(0.020) & \\ 0.019(0.007) & 0.015(0.004) \end{bmatrix}$$

$$\begin{bmatrix} e_{0ij} \end{bmatrix} \sim \text{N}(0,\ \Omega_e) \ : \ \Omega_e = \begin{bmatrix} 0.554(0.013) \end{bmatrix}$$

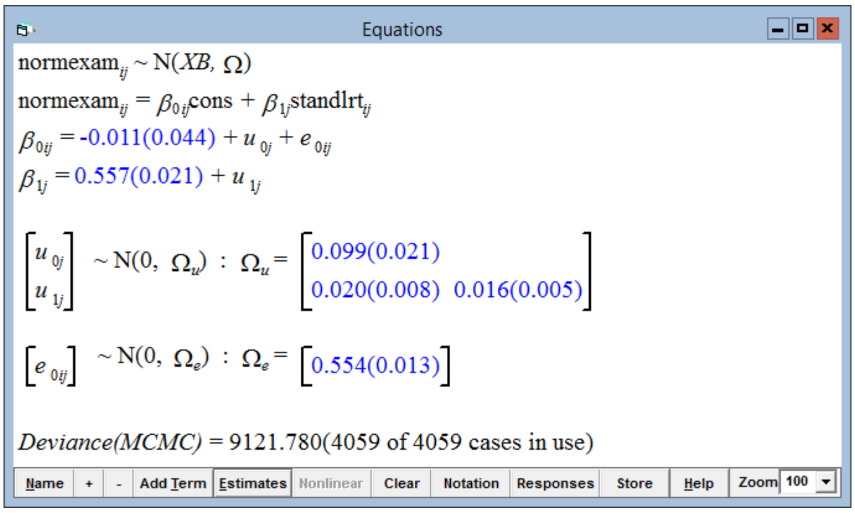
Deviance(MCMC) = 9122.987(4059 of 4059 cases in use)

If we now want to use parameter expansion we need to do the following:

- Click on the **IGLS/RIGLS** tab on the **Estimation control** window.
- Click **Start**.

- Click on the **Estimation Control** button.
- Select the tab labelled **MCMC** on the **Estimation control** window.
- Select **MCMC/MCMC options** from the **Model** menu.
- Click on the level **2** tick box under **Parameter expansion at level:**.
- Click on the **Done** button on the **MCMC options** window.
- Click **Start**.

This will give the following output in the **Equations** window:



$$\text{normexam}_{ij} \sim \text{N}(XB, \Omega)$$
$$\text{normexam}_{ij} = \beta_{0ij}\text{cons} + \beta_{1j}\text{standlrt}_{ij}$$
$$\beta_{0ij} = -0.011(0.044) + u_{0j} + e_{0ij}$$
$$\beta_{1j} = 0.557(0.021) + u_{1j}$$
$$\begin{bmatrix} u_{0j} \\ u_{1j} \end{bmatrix} \sim \text{N}(0, \Omega_u) : \Omega_u = \begin{bmatrix} 0.099(0.021) & \\ 0.020(0.008) & 0.016(0.005) \end{bmatrix}$$
$$\begin{bmatrix} e_{0ij} \end{bmatrix} \sim \text{N}(0, \Omega_e) : \Omega_e = \begin{bmatrix} 0.554(0.013) \end{bmatrix}$$

Deviance(MCMC) = 9121.780(4059 of 4059 cases in use)

Here we see that the level 2 variance matrix estimates have all increased slightly with the new prior induced by parameter expansion. In the table below we compare ESS values for the two approaches:

Parameter	Normal Estimate	Normal ESS	P.E. Estimate	P.E. ESS
β_0	-0.006 (0.039)	281	-0.011 (0.044)	215
β_1	0.558 (0.020)	806	0.557 (0.021)	697
σ^2_{u0}	0.096 (0.020)	2937	0.099 (0.021)	2768
σ_{u01}	0.019 (0.007)	1811	0.020 (0.008)	1771
σ^2_{u1}	0.015 (0.004)	1170	0.016 (0.005)	978
σ^2_e	0.554 (0.013)	5250	0.554 (0.013)	4594

Interestingly here all the ESS values are slightly worse for the parameter expanded formulation although not particularly different. This may just be because, as we saw for the variance components model, parameter expansion doesn't have much impact for models with significant cluster variability. It may also be interesting to investigate fitting different α terms for the random intercepts and slopes. This would be possible (and could be easily done in WinBUGS) and again the original parameters will be recoverable but we leave this for future research.

Chapter learning outcomes

- ⋆ How and when to use parameter expansion to improve mixing.
- ⋆ The use of parameter expansion in Normal and binomial models.
- ⋆ The use of parameter expansion with random slopes.
- ⋆ How to use parameter expansion and the WinBUGS interface.

Chapter 25

Hierarchical Centring

In this chapter we consider another technique that aims to improve the mixing of MCMC algorithms via reparameterisation. Hierarchical centring (Gelfand et al., 1995) is the third technique that focuses on the correlation between fixed effects and residuals. While SMCMC updates the parameters together in one block and the structured MVN approach integrates out the residuals, hierarchical centring reparameterises the model by replacing the residuals by other terms. The advantage hierarchical centring has over the other two approaches (as implemented in MLwiN) is that it can be used for non-Normal responses. In the sections that follow we will introduce the method in more detail and go through examples of its use with several response types. We will show how it can be used for Normal responses via the WinBUGS interface and how we have implemented a version for non-Normal responses directly in MLwiN. We finish by describing an approach for Normal responses that follows on from the non-Normal implementation but which we have not verified as a true MCMC algorithm.

25.1 What is hierarchical centering?

Let us begin once again with our favourite 2-level **tutorial** example which has as a response exam marks of pupils clustered as they are taught in groups in a set of schools.

Let y_{ij} be the mark for pupil i in school j; then perhaps the simplest multilevel model we could fit is

$$y_{ij} = \beta_0 + u_j + e_{ij}, \quad u_j \sim \mathrm{N}(0, \sigma_u^2), \quad e_{ij} \sim \mathrm{N}(0, \sigma_e^2)$$

Here we have an estimated average mark, β_0, school level residuals, u_j, with variance σ_u^2, and pupil level residuals, e_{ij}, with variance σ_e^2.

For a Bayesian model we will use ‘diffuse’ priors as follows:

$$p(\beta_0) \propto 1, \quad p(1/\sigma_u^2) \sim \Gamma(\varepsilon, \varepsilon), \quad p(1/\sigma_e^2) \sim \Gamma(\varepsilon, \varepsilon)$$

As described in Chapter 21 a four step Gibbs sampling algorithm can be used for this model and is the default in MLwiN. This formulation is often described as the ‘uncentred’ formulation as the random effects are centred around (have mean) 0 rather than around a function of the fixed effects.

In the case of the above model a natural alternative parameterisation would be to centre the random effects around the overall mean β_0. This is known as hierarchical centering and can be achieved by replacing u_j with $u_j^* = \beta_0 + u_j$ and then the model is written:

$$y_{ij} = u_j^* + e_{ij}, \quad u_j^* \sim \mathrm{N}(\beta_0, \sigma_u^2), \quad e_{ij} \sim \mathrm{N}(0, \sigma_e^2)$$

To fit this parameterisation we still have a four step Gibbs sampling algorithm, however we now have a new step for β_0 conditioning on u_j^* and a step for u_j^* that replaces the current step for u_j. The fixed effect β_0 will have exactly the same meaning and the original u_j can be retrieved by calculating $u_j = u_j^* - \beta_0$. This algorithm will work better than the original if the correlation between u_j^* and β_0 is of smaller magnitude than the correlation between u_j and β_0 which is usually the case provided the cluster variance σ_u^2 is an appreciable part of the total variance.

Centring is not limited to simply centring around an overall mean or intercept as any higher level predictors can be moved up the hierarchy. It is also possible with random slopes to move the corresponding fixed effects up the hierarchy so for example the following model

$$\begin{gathered} y_{ij} = \beta_0 + \beta_1 x_{1ij} + \beta_2 x_{2j} + \beta_3 x_{3ij} + u_{0j} + u_{1j} x_{1ij} + e_{ij}, \\ u_j \sim \mathrm{N}(0, \Omega_u), \quad e_{ij} \sim \mathrm{N}(0, \sigma_e^2) \end{gathered}$$

could be reparameterised as follows:

$$\begin{gathered} y_{ij} = \beta_3 x_{3ij} + u_{0j}^* + u_{1j}^* x_{1ij} + e_{ij}, \\ \begin{pmatrix} u_{0j}^* \\ u_{1j}^* \end{pmatrix} \sim \mathrm{N}\left(\begin{pmatrix} \beta_0 + \beta_2 x_{2j} \\ \beta_1 \end{pmatrix}, \begin{pmatrix} \sigma_{u0}^2 & \sigma_{u01} \\ \sigma_{u01} & \sigma_{u1}^2 \end{pmatrix} \right), \quad e_{ij} \sim \mathrm{N}(0, \sigma_e^2) \end{gathered}$$

This example illustrates how complicated the reparameterisation can be. Here we have moved three of the fixed effects up the hierarchy but obviously other possible parameterisations exist where for example any permutation of the three fixed effects are moved. If we have a three level nested structure things get even more complex as it is feasible as is done in Gelfand et al. (1995) to centre the level 2 residuals around the level 3 residuals and the level 3 residuals around the fixed effects.

For the implementation of hierarchical centering in MLwiN we only allow centring for one chosen classification (whether the model is nested or crossed) and the random effects for this classification will be centred (where possible) around the fixed effects. The implementation will also perform all feasible centring operations as described in the second example above without giving the user options with regard what to centre.

For the Gibbs sampling algorithm used by default for Normal models in MLwiN we have not implemented the hierarchical centering algorithm directly and so as we see in the next section we rely on the WinBUGS interface for these models. Note that the hierarchical centering algorithm with Metropolis steps for the fixed and random effects is available for Normal models as described for non-Normal models and an alternative algorithm will be described later.

25.2 Centring Normal models using WinBUGS

We will first set up a variance components model using the **tutorial** dataset. Note we already looked at this model using WinBUGS in chapter 7.

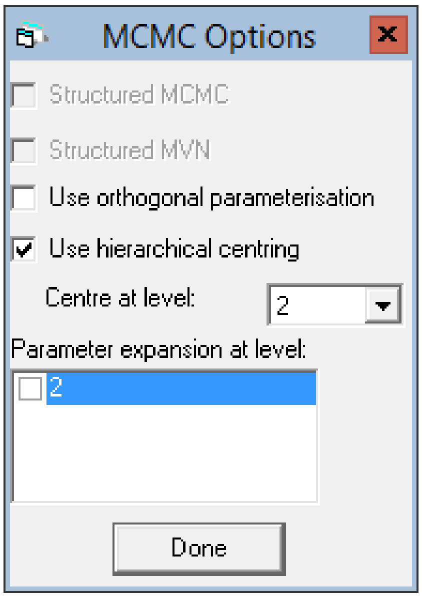
- Select **Open sample worksheet** from the **File** menu.
- Select **tutorial.ws**.
- Select **Equations** from the **Model** menu.
- Click on **y** (either of the **y** symbols shown will do).
- In the **y** list, select **normexam**.
- In the **N levels** list, select **2-ij**.
- In the **level 2(j):** list, select **school**.
- In the **level 1(i):** list, select **student**.
- Click on the **Done** button.
- Click on the red x_0.
- In the drop-down list, select **cons**.
- Check the box labelled **i(student)**.
- Check the box labelled **j(school)**.
- Click on the **Done** button.
- Click the **Add Term** button on the **Equations** window tool bar.

- Select **standlrt** from the **variable** list.
- Click on the **Done** button.
- Click **Start**.

The above instructions will run the model using IGLS. We next need to change to MCMC and set up the hierarchical centering options.

- Select the tab labelled **MCMC** on the **Estimation control** window.
- Select **MCMC/MCMC options** from the **Model** menu.
- Click on the **Hierarchical centering** tick box.
- Select 2 from the **Centre at level** list.

The **MCMC options** window will then look as follows:


MCMC Options

☐ Structured MCMC

☐ Structured MVN

☐ Use orthogonal parameterisation

☑ Use hierarchical centring

Centre at level: 2

Parameter expansion at level:

☐ 2

We next need to confirm these settings and write out the model to WinBUGS:

- Click on the **Done** button.
- Select **MCMC/WinBUGS options** from the **Model** menu.
- Click on the **WinBUGS 1.4** button.
- Click on the big button at the top of the **WinBUGS options** window.
- Save the file as **hc1.bug** in the window that appears.

We will then start up WinBUGS and look at the code that appears. We will need to read in the file **hc1.bug** (from the directory it was saved in) as a text file. To do this we will have to change the **Files of type** box to **All files (*.*)** to see the file **hc1.bug**. Having read in the file, a window headed **hc1.bug** will appear containing the information needed by BUGS for this model.

The model code for this model is fairly short and is detailed below:

```
# WINBUGS 1.4 code generated from MLwiN program


#----MODEL Definition----------------


model
{
# Level 1 definition
for(i in 1:N) {
normexam[i] ~ dnorm(mu[i],tau)
mu[i]<- beta[2] * standlrt[i]
+ u2[school[i]] * cons[i]
}
# Higher level definitions
for (j in 1:n2) {
u2[j] ~ dnorm(mu.u2[j],tau.u2)
}
# Priors for fixed effects
for (k in 1:2) { beta[k] ~ dflat() }
# Priors for random terms
tau ~ dgamma(0.001000,0.001000)
sigma2 <- 1/tau
for (i in 1:n2)
{
mu.u2[i]<-beta[1] * hccons[i]
}
tau.u2 ~ dgamma(0.001000,0.001000)
sigma2.u2 <- 1/tau.u2
}
```

Here we see that centring has been performed in the code with **mu.u2[j]** defining the mean of school residual j and **mu.u2[i]** being defined as **beta[1]**$\times$ **hccons[i]** lower down in the code. Basically **hccons** is a constant vector of length 65 (the number of schools).

We will now run the model and store the results in WinBUGS and then read in the output files into MLwiN. Before running a model in WinBUGS we first need to read in the particular elements of the model using the **Specification** window available from the **Model** menu. After selecting the window containing the model by clicking on it, clicking on the **check model** button should give the message 'model is syntactically correct' at the bottom of the screen. Next we need to load in the data for the model. Due to the fact that the data is generally the largest part of the file generated by MLwiN it is included after the initial values. The data section always begins as follows:

```
#----Data File---------------------------------
list(
```

To load the data into BUGS we need to highlight the **list** identifier at the start of the data list and click on the **load data** button in the **specification** window. If this is successful the message 'data loaded' will appear at the bottom of the screen. Next we have to combine the data and model definition by clicking on the **compile** button. Again if this operation is successful a message appears at the bottom of the screen, this time stating 'model compiled'. Finally as BUGS uses MCMC methods all unknown parameters will need starting values. These are included in the initial values part of the file that starts as follows:

```
#----Initial values file----------------------------

list
```

To use these values in WinBUGS we need to highlight the **list** identifier at the start of the initial values and click on the **load inits** button on the **specification** window. This will then give the final message 'model is initialized'.

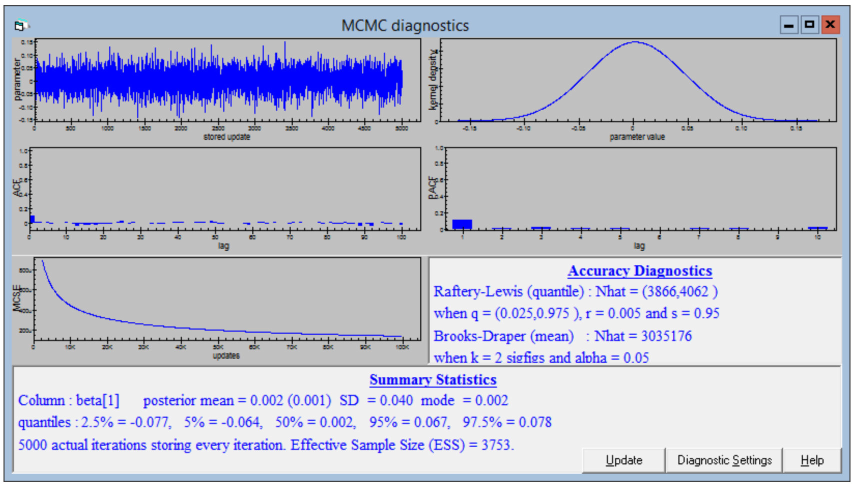
Before we start we have to tell WinBUGS which parameters we wish to monitor. We will here choose several parameters. From the **Inference** menu select the **Samples** options and a window will appear that allows the user to specify which parameters to monitor. In this window we will firstly select the fixed effects by typing **beta** in the **node** box. Note that when a correctly typed parameter is input the **set** button will become enabled. We will also want to use a burn-in of some iterations. For this we will use 500 iterations as is used in MLwiN so we will also modify the **beg value** from **1** to **501**. After this press the **set** button and the parameter will be set for monitoring. We now need to repeat this procedure with the two variance parameters **sigma2.u2** and **sigma2**.

We are now ready to set the estimation engine running and this is done via the **Update** window found in the **Model** menu. We need to specify the number of updates (including the burn-in) and so we will replace the **1000** here with **5500** to give 5000 iterations after the burn-in as is used in MLwiN. We then press the **update** button to start the sampler.

After running the model in WinBUGS we will bring the chains back into MLwiN for further analysis. WinBUGS also has the option to produce input files in a format originally for a package called CODA. MLwiN can also use these files to input the parameter chains from WinBUGS into columns in MLwiN. Here we will consider all parameters stored by using the * option so select this in the **node** box and press the **coda** button on the **sample** window. This will produce two windows that are labelled **CODA index** which contains the variable names and **CODA for chain 1** which contains the values for the parameter chains. We will now save these files as text files by clicking on the respective windows and then choosing **Save As** from the **File** menu. We will need to save the files in *plain text* (*.txt) format. We will store the **CODA index** file as **hc1.ind** and the **CODA for chain 1** file as **hc1.out** in the same directory as **hc1.bug**. Note that these are the extensions that the classic BUGS used for these files but, as we have selected the plain text format, WinBUGS will add an additional '.txt' to the first filename and so the files are actually saved as **hc1.ind.txt** and **hc1.out.txt**.

Now back in MLwiN if you want to input the traces return to the **BUGS options** window that we used earlier (available from the **Model** menu). Here we will need to modify the **.out** and **.in** file fields to **hc1.out** and **hc1.ind.txt** respectively. Note that if you did not put these files in the current directory you will have to include their full path names in the respective boxes. Pressing the **Input data** button will now load the chains into columns **c300** to **c303**.

We can now use the MLwiN MCMC diagnostics on the BUGS output, for example if we want to look at the β_0 chain labelled as **beta[1]** in the column list we see the following:



Accuracy Diagnostics
Raftery-Lewis (quantile) : Nhat = (3866,4062)
when q = (0.025,0.975), r = 0.005 and s = 0.95
Brooks-Draper (mean) : Nhat = 3035176
when k = 2 sigfigs and alpha = 0.05

Summary Statistics
Column : beta[1] posterior mean = 0.002 (0.001) SD = 0.040 mode = 0.002
quantiles : 2.5% = -0.077, 5% = -0.064, 50% = 0.002, 95% = 0.067, 97.5% = 0.078
5000 actual iterations storing every iteration. Effective Sample Size (ESS) = 3753.

Here we see a reasonable ESS and in the table below we compare hierarchical centering with the other previously considered techniques for this model.

Parameter	Gibbs	SMCMC	SMVN	HC
β_0	216	5209	1125	3753
β_1	4413	5332	1219	4207
σ^2_{u0}	2821	3108	1015	2873
σ^2_{e0}	4712	4707	1276	4810

As we might expect the only parameter that changes ESS greatly when compared with the standard (uncentred) Gibbs algorithm is the intercept β_0 as this is the only parameter of the four directly affected by centring. The HC method doesn't do quite as well as SMCMC but is more generalisable to other models as we see next.

25.3 Binomial hierarchical centering algorithm

The technique of hierarchical centering can also be used in any model which contains a linear predictor with random effects. This class of models includes all the other types of multilevel models, for example binomial, Poisson and multinomial models in MLwiN. The idea in these models is as with the Normal response models pushing the fixed effects that are constant across clusters up the hierarchy.

We will see, as described in Browne et al. (2009*b*), that for these models where a random walk Metropolis algorithm is used the changes required to the algorithm for hierarchical centering are very small. We will consider again the dataset taken from the 1988 Bangladesh Fertility Survey which contains 1934 women nested within 60 districts. The response variable y_{ij} is a binary indicator of whether or not woman i in district j was using contraceptives at the time of the survey. We begin by considering the simple model with only an intercept term and district-level random effects. A two-level variance components model for $\pi_{ij} = Pr(y_{ij} = 1)$ can be written

$$\begin{gathered} y_{ij} \sim \text{ Bernoulli}(\pi_{ij}) \\ \text{logit}(\pi_{ij}) = \beta_0 + u_j, \quad u_j \sim \text{N}(0, \sigma_u^2) \\ p(\beta_0) \propto 1, \quad p(\sigma_u^2) \sim \Gamma^{-1}(\varepsilon, \varepsilon) \end{gathered}$$

where β_0 is the fixed intercept, and u_j are random district effects with variance σ_u^2. We assume diffuse priors and use an improper uniform prior for β_0 and a commonly used (and conjugate) inverse Gamma prior for σ_u^2. To fit this model the standard algorithm used in MLwiN consists of the following:

Step 1: Update β_0 using random walk Metropolis sampling.

Step 2: Update $u_j, \quad j = 1, \ldots, 60$ using random walk Metropolis sampling.

Step 3: Update σ_u^2 from its inverse Gamma full conditional using Gibbs sampling.

We can reparameterise the above model by replacing the residuals u_j by the random effects $u_j^* = \beta_0 + u_j$ which leads to the following model formulation:

$$\begin{aligned} y_{ij} &\sim \text{Bernoulli}(\pi_{ij}) \\ \text{logit}(\pi_{ij}) &= u_j^*, \quad u_j^* \sim \text{N}(\beta_0, \sigma_u^2) \\ p(\beta_0) &\propto 1, \quad p(\sigma_u^2) \sim \Gamma^{-1}(\varepsilon, \varepsilon) \end{aligned}$$

Here the u_j^* are (hierarchically) centred around β_0. Using this parameterisation we have different conditional distributions as we have replaced u_j with u_j^* and now we can construct an MCMC algorithm with conjugate Gibbs sampler steps for both β_0 and σ_u^2 while keeping random walk Metropolis steps for the 60 u_j^*.

The step for updating β_0 from its full conditional distribution is as follows:

$$\begin{aligned} p(\beta_0 \mid y, u^*, \sigma_u^2) &\sim \text{N}(\widehat{\beta}_0, \widehat{D}), \quad \text{where} \\ \widehat{\beta}_0 &= \textstyle\sum_{j=1}^{60} \frac{u_j^*}{60} \quad \text{and} \quad \widehat{D} = \frac{\sigma_u^2}{60}, \end{aligned}$$

Here we see that the update step now only involves the 60 random effects and not the full 1934 data points. Interestingly we can consider this step (at iteration $t+1$) in terms of the original parameters as follows:

$$\begin{aligned} p(\beta_0 \mid y, u^*, \sigma_u^2) &\sim \text{N}(\widehat{\beta}_0, \widehat{D}), \quad \text{where} \\ \widehat{\beta}_0 &= \beta_0^{(t)} + \textstyle\sum_{j=1}^{60} \frac{u_j}{60} \quad \text{and} \quad \widehat{D} = \frac{\sigma_u^2}{60}, \end{aligned}$$

where $\beta_0^{(t)}$ is the current value of β_0 (at iteration t).

If we are considering the original parameterisation then we must respect that the above is still conditional on u_j^* and not u_j and so after updating β_0 we have

$$u_j = u_j^{(t)} + \beta_0^{(t)} - \beta_0^{(t+1)} \quad \forall j = 1, \ldots, J$$

to ensure that u_j^* is kept fixed, where $u_j^{(t)}$ is the value of u_j prior to updating β_0.

The new random walk Metropolis step for u_j^* when considered as a step for u_j is identical to the Metropolis step in the original algorithm — we are conditioning on β_0 and as $u_j = u_j^* - \beta_0$, proposing an additive jump for u_j^* induces the same additive jump in u_j.

This means that, from an algorithm programming point of view, to transform the standard algorithm for the first model we simply need to modify the step for β_0 to a Gibbs sampling step with an additional correction to the u_j and leave the other two steps alone.

This has been implemented in MLwiN and we show this in practice in the next section. What should be noted for non-Normal models is that hierarchical centering not only benefits from potentially having less correlated parameters but also the fact that some Metropolis steps are replaced by quicker Gibbs sampling steps that often themselves result in better mixing.

25.4 Binomial example in practice

We will here consider the model that we looked at in Chapter 23. We will now repeat the instructions on how to set up this model in MLwiN.

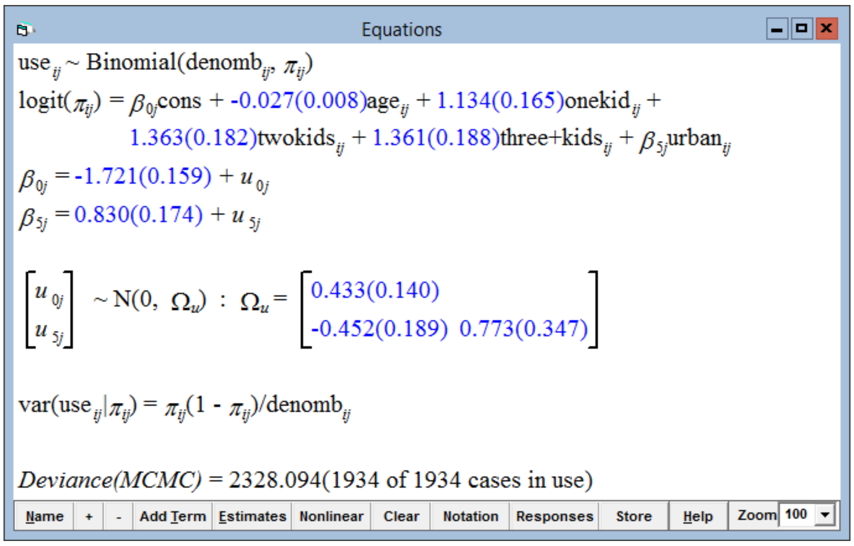
- Select **Open sample worksheet** from the **File** menu.
- Select **bang1.ws** from the list of worksheets.
- Select **Open**.
- Select **Equations** from the **Model** menu.
- Click on the **Clear** button to remove any existing model in the worksheet
- Click on the red y.
- Select **use** for the **y** variable.
- Select **2-ij** for the **number (N) of levels**.
- Select **district** as **level 2(j)**.
- Select **woman** as **level 1(i)**.
- Click on the **Done** button.
- Click on the **N** and instead choose **Binomial** from the list.
- Click on the **Done** button.
- Click on the red n_{ij}.
- Select **denomb** and click on the **Done** button.
- Click on the red x_0.
- Select **cons** and click on the **Done** button.
- Click on **cons** in the **Equations** window.
- Click on the **(j)district** box in the **X variable** window.
- Click on the **Done** button. (Note the level 1 variance is Binomial and so doesn't need adding).
- Click on the **Add Term** button.
- Select **age** from the **variable** list and click on the **Done** button.
- Click on the **Add Term** button on the **Equations** window.
- Select **lc** from the **variable** pull-down list.

- Click on the **Done** button.
- Click on the **Add Term** button on the **Equations** window.
- Choose **urban** from the **Variable** list.
- Click on the **Done** button.
- Click on the **urban** predictor.
- In the **X variable** window click in the **j(district)** tickbox.
- Click on the **Done** button.
- Click on the **Start** button.

In chapter 23 we compared the reparameterisation with the standard method. Here we will move straight to using hierarchical centering

- Select the tab labelled **MCMC** on the **Estimation control** window.
- Select **MCMC/MCMC options** from the **Model** menu.
- Click on the **Hierarchical Centring tick box**.
- Select 2 from the **Centre at level** list.
- Click on the **Done** button.
- Click on the **Start** button.

The **Equations** window will look as follows:



$\text{use}_{ij} \sim \text{Binomial}(\text{denomb}_{ij}, \pi_{ij})$

$\text{logit}(\pi_{ij}) = \beta_{0j}\text{cons} + -0.027(0.008)\text{age}_{ij} + 1.134(0.165)\text{onekid}_{ij} + 1.363(0.182)\text{twokids}_{ij} + 1.361(0.188)\text{three+kids}_{ij} + \beta_{5j}\text{urban}_{ij}$

$\beta_{0j} = -1.721(0.159) + u_{0j}$

$\beta_{5j} = 0.830(0.174) + u_{5j}$

$$\begin{bmatrix} u_{0j} \\ u_{5j} \end{bmatrix} \sim \text{N}(0,\ \Omega_u) \ :\ \Omega_u = \begin{bmatrix} 0.433(0.140) & \\ -0.452(0.189) & 0.773(0.347) \end{bmatrix}$$

$\text{var}(\text{use}_{ij}|\pi_{ij}) = \pi_{ij}(1 - \pi_{ij})/\text{denomb}_{ij}$

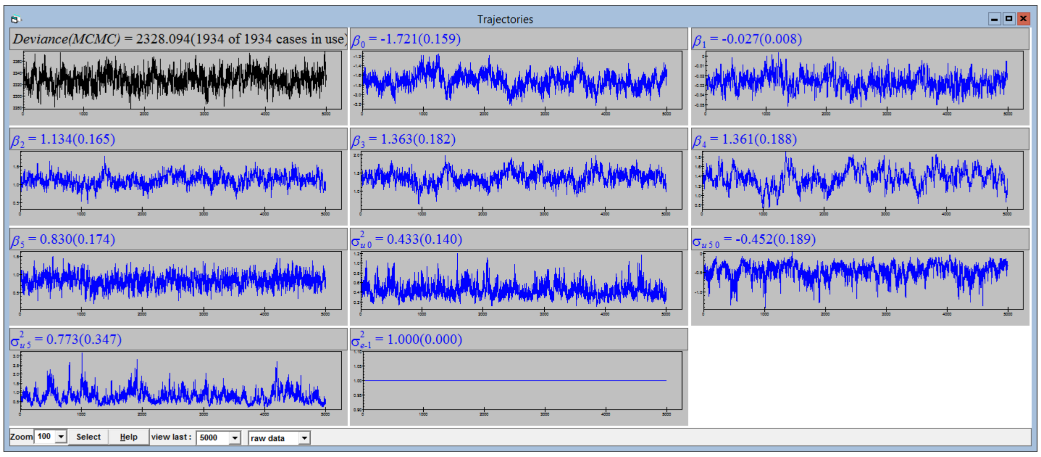
Deviance(MCMC) = 2328.094(1934 of 1934 cases in use)

We will need to look at the Trajectories window to examine the mixing of the parameters:

- Select **Trajectories** from the **Model** menu.

- Modify the **view last** box to **5000**.
- Click on the **select** button.
- In the pull down list choose **3 graphs per row**.
- Click on the **Done** button.

The **Trajectories** window will look as follows:



Deviance(MCMC) = 2328.094(1934 of 1934 cases in use)

β_0 = -1.721(0.159)

β_1 = -0.027(0.008)

β_2 = 1.134(0.165)

β_3 = 1.363(0.182)

β_4 = 1.361(0.188)

β_5 = 0.830(0.174)

σ^2_{u0} = 0.433(0.140)

σ_{u50} = -0.452(0.189)

σ^2_{u5} = 0.773(0.347)

σ^2_{e1} = 1.000(0.000)

Here we see that mixing is generally similar to that seen for the standard algorithm, although the mixing for the β_5 parameter has improved (we give ESS values in the table later). The hierarchical centring steps will affect the mixing of parameters β_0 and β_5 only due to the random intercepts and slopes and the fact that no other predictors in this model are constant at the higher level.

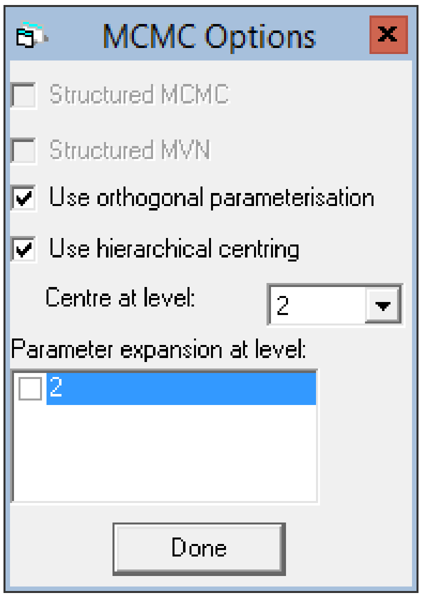
We can also look at combining the two reparameterisation methods, orthogonal predictors and hierarchical centring, as is considered in Browne et al. (2009*b*). Here the order of predictors matters as the fixed predictors will be transformed. In the current order only the intercept parameter will be centred (as it is first it will be replaced by itself in the orthogonal set created) as the **urban** vector will be replaced by an orthogonal vector which can then not be centred around the slope residuals.

To use both methods we need to do the following:

- Click on the **IGLS/RIGLS** tab on the **Estimation control** window.
- Click **Start**.
- Click on the **Estimation Control** button
- Select the tab labelled **MCMC** on the **Estimation control** window.
- Select **MCMC/MCMC options** from the **Model** menu.

- Click on **Use orthogonal parameterisation**.

The **MCMC options** window should then look as follows:


MCMC Options

☐ Structured MCMC

☐ Structured MVN

☑ Use orthogonal parameterisation

☑ Use hierarchical centring

Centre at level: 2

Parameter expansion at level:

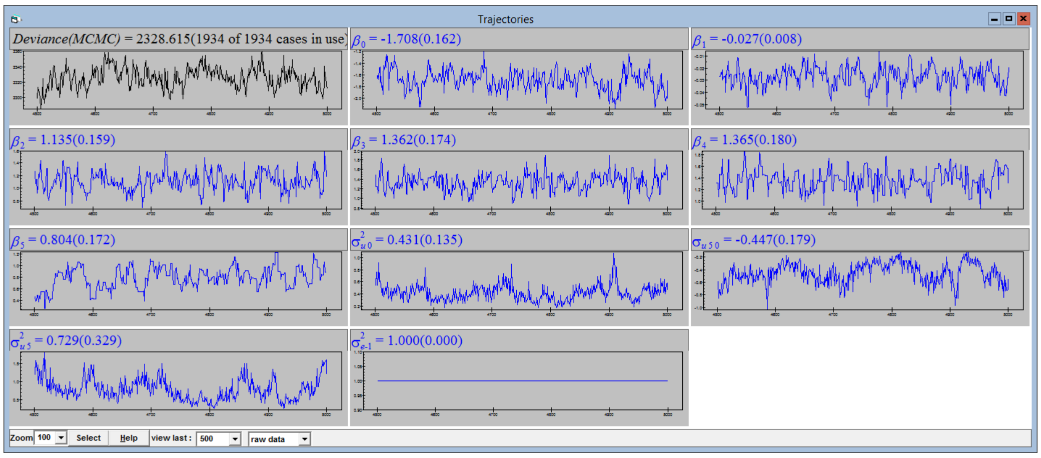
☐ 2

Done

To next run the model using both the orthogonal parameterisation and hierarchical centring we do the following:

- Click on the **Done** button on the **MCMC options** window.
- Click **Start**.

After running the **Trajectories** window will look as follows:


Trajectories

Deviance(MCMC) = 2328.615(1934 of 1934 cases in use)

β_0 = -1.708(0.162)

β_1 = -0.027(0.008)

β_2 = 1.135(0.159)

β_3 = 1.362(0.174)

β_4 = 1.365(0.180)

β_5 = 0.804(0.172)

σ^2_{u0} = 0.431(0.135)

σ_{u50} = -0.447(0.179)

σ^2_{u5} = 0.729(0.329)

σ^2_{e1} = 1.000(0.000)

Here we see in fact the best trace for β_0 of all the methods considered. The table below shows ESS values for all four methods considered in this chapter and the orthogonal chapter.

Parameter	Standard	Orthogonal	HC	HC + Ort
β_0	73	301	53	507
β_1	190	1154	165	1124
β_2	183	977	100	1111
β_3	143	1024	67	1037
β_4	75	924	52	1048
β_5	60	124	435	257
σ^2_{u0}	229	228	193	259
σ_{u50}	148	136	105	138
σ^2_{u5}	130	116	93	92

Here we see that changing to an orthogonal parameterisation has greater impact for this model than hierarchical centring. If we altered the order of predictors and put **urban** first then we would get hierarchical centring of slopes but not intercepts which gives a better ESS for β_5 but worse for β_0.

This model is simply one example and we can look at simpler examples where hierarchical centring has more of an impact. For example if we were to instead look at a variance components model with no predictors except the intercept (this can be set up by deleting the other predictors via the **Equations** window) then we can compare the effect of centring against not centring. Note for this model there is only one fixed effect so there is no difference between the standard and orthogonal parameterisations.

The results of running this model using both the centred and uncentred parameterisations are given in the following table:

Parameter	Standard	HC
β_0	221	1194
σ^2_{u0}	430	522

Here we see a 5-fold increase in ESS for β_0 showing the benefit of centring for this model. We will next move on to reconsider a Poisson model that we also considered in the orthogonal parameterisation chapter.

25.5 The Melanoma example

In chapter 23 we revisited a Poisson response model for a dataset that concerned Melanoma mortality in Europe. The model we considered had fixed effects and interactions for nation and UV exposure and we will revisit this model again here while using hierarchical centring. First we need to set up the model in MLwiN and hence we need to follow the following instructions as in Chapter 23.

- Select **Open sample worksheet** from the **File** menu.
- Select **mmmec.ws** and click on **Open**.
- Open the **Command interface** window from the **Data Manipulation** menu and enter the following commands:

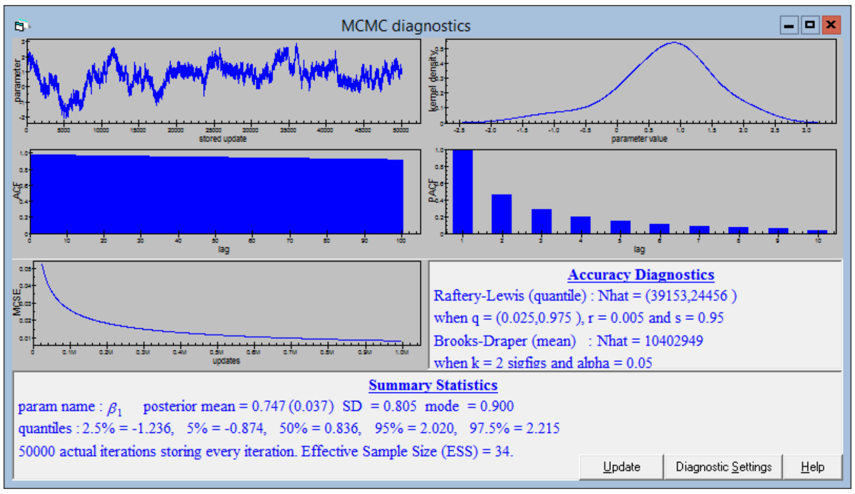
```
▶ calc c9 = loge('exp')
▶ name c9 'logexp'
```

- Select **Equations** from the **Model** menu.
- Click on the red y.
- Select **obs** as the y variable.
- Select **2-ij** as the number of levels.
- Select **region** as **level 2(j)**.
- Select **county** as **level 1(i)**.
- Click on the **Done** button.
- Click on the **N** and select **Poisson** from the popup list and click on the **Done** button.
- Click on the π_{ij} and from the **offset** window that appears select **logexp**.
- Click on the **Done** button.
- Click on the red x_0 and select **cons** and click on the **Done** button.
- In the **Equations** window click on β_0 (**cons**).
- Click in the **(j)region** tick box.
- Remove the tick in the **Fixed parameter** tick box.
- Click on the **Done** button.
- Click on the **Add Term** button on the **Equations** window.
- Select **nation** from the **variable** dropdown list.
- Select [**none**] from the **reference category** dropdown list.
- Click on the **Done** button.
- Click on the **Add Term** button on the **Equations** window.
- Select **1** in the **order** box.
- Select **nation** from the first **variable** dropdown list.
- Select **uvbi** from the second **variable** dropdown list.
- Select [**none**] from the first **ref cat** dropdown list.
- Click on the **Done** button.
- Click on the **Start** button.

The above list of instructions should result in setting up the model structure for the model originally introduced in section 11.4 and fitting the model in IGLS. We next need to change estimation method to MCMC and ask for hierarchical centring before fitting the model. As in chapter 23 we will also need to increase the run length because of the potential poor mixing of the chains for this model. This we do as follows:

- Click on the **Estimation Control** button.
- Click on the **MCMC** tab.
- Change the **Monitoring chain length** to **50,000**.
- Change the **refresh rate** to **500**.
- Click on the **Done** button.
- Select **MCMC/MCMC options** from the **Model** menu.
- Click on the **Hierarchical Centring tick box**.
- Select 2 from the **centre at level** list.
- Click on the **Done** button.
- Click on the **Start** button.

If you look closely at the **Equations** window you will see that all the nation predictors have simply a j subscript. This is because regions are nested within nations and consequently the nation variables will be a constant at the region level. This means that when we choose hierarchical centring we will push all 9 nation effects up the hierarchy. After running for 50,000 iterations we can, as in Chapter 23, look at the chain for the Belgium effect, β_1 (via the **Trajectories**) window; we see the following:



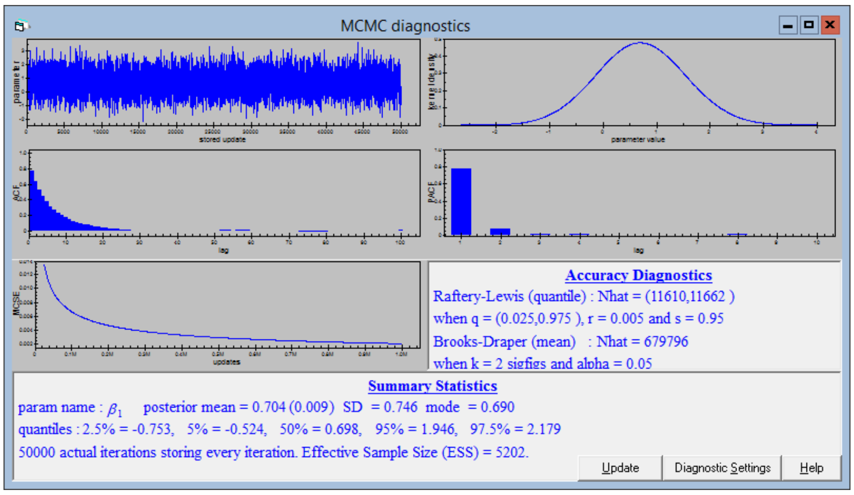
Accuracy Diagnostics
Raftery-Lewis (quantile) : Nhat = (39153,24456)
when q = (0.025,0.975), r = 0.005 and s = 0.95
Brooks-Draper (mean) : Nhat = 10402949
when k = 2 sigfigs and alpha = 0.05

Summary Statistics
param name : β_1 posterior mean = 0.747 (0.037) SD = 0.805 mode = 0.900
quantiles : 2.5% = -1.236, 5% = -0.874, 50% = 0.836, 95% = 2.020, 97.5% = 2.215
50000 actual iterations storing every iteration. Effective Sample Size (ESS) = 34.

Here we see that there is very little improvement from using hierarchical centring, as the main cause of the poor mixing is the correlation between the pairs of predictor and interaction for each nation. We can, as we did for the last example, look at using both hierarchical centring and an orthogonal parameterisation. This will still centre the first 9 nation effects as they are already orthogonal but will replace the interactions with orthogonal terms. This can be achieved as follows:

- Click on the **IGLS/RIGLS** tab on the **Estimation control** window.
- Click **Start**.
- Click on the **Estimation Control** button
- Select the tab labelled **MCMC** on the **Estimation control** window.
- Select **MCMC/MCMC options** from the **Model** menu.
- Click on **Use orthogonal parameterisation**.
- Click on the **Done** button on the **MCMC options** window.
- Click **Start**.

Again if we look at the trace for β_1 as shown below



Accuracy Diagnostics
Raftery-Lewis (quantile) : Nhat = (11610,11662)
when q = (0.025,0.975), r = 0.005 and s = 0.95
Brooks-Draper (mean) : Nhat = 679796
when k = 2 sigfigs and alpha = 0.05

Summary Statistics
param name : β_1 posterior mean = 0.704 (0.009) SD = 0.746 mode = 0.690
quantiles : 2.5% = -0.753, 5% = -0.524, 50% = 0.698, 95% = 1.946, 97.5% = 2.179
50000 actual iterations storing every iteration. Effective Sample Size (ESS) = 5202.

we see as we saw in Chapter 23 that for this parameter the chain mixes much better. To give a fair comparison of the whole model we give ESS for all parameters in the following table:

Parameter	Standard	Orthogonal	HC	HC + Orth
β_1	40	4905	34	5202
β_2	178	650	254	1386
β_3	38	5315	33	5653
β_4	1697	1764	7891	12101
β_5	73	1005	150	1224
β_6	410	1640	475	2402
β_7	88	6770	44	6858
β_8	27	2220	26	1581
β_9	29	2360	32	2487
β_{10}	40	5104	33	5155
β_{11}	211	1052	181	1057
β_{12}	38	5307	33	5582
β_{13}	1769	1964	1767	1896
β_{14}	77	1106	131	1084
β_{15}	464	1764	388	1781
β_{16}	87	6643	44	6266
β_{17}	27	2190	26	1560
β_{18}	29	2352	31	2410
σ^2_{u0}	4132	4248	6953	6766

Here we see that, although it is slight, there does seem to be some additional benefit of using hierarchical centring on top of the orthogonal parameterisation as for the ESS increases for the worst mixing parameters. It should also be noted that both the non-Normal models were chosen as they illustrated improvements when using orthogonal parameterisations. If for example we consider the simpler model with only nation effects and no interactions then the original parameters are already orthogonal and so we can only compare the standard formulation and a centred formulation. Here the improvement of using hierarchical centring is very impressive as shown in the table below:

Parameter	Non-centred	Centred
β_1	627	29864
β_2	345	31700
β_3	413	34555
β_4	1206	21562
β_5	468	30527
β_6	1196	17241
β_7	3493	7002
β_8	3221	11419
β_9	589	27128
σ^2_{u0}	7262	11818

We have here seen that generally hierarchical centring results in a better MCMC algorithm for non-Normal response models. One note of caution as demonstrated in Browne et al. (2009*b*) in one example is that if the cluster variance is small, once again hierarchical centring can result in worse chains.

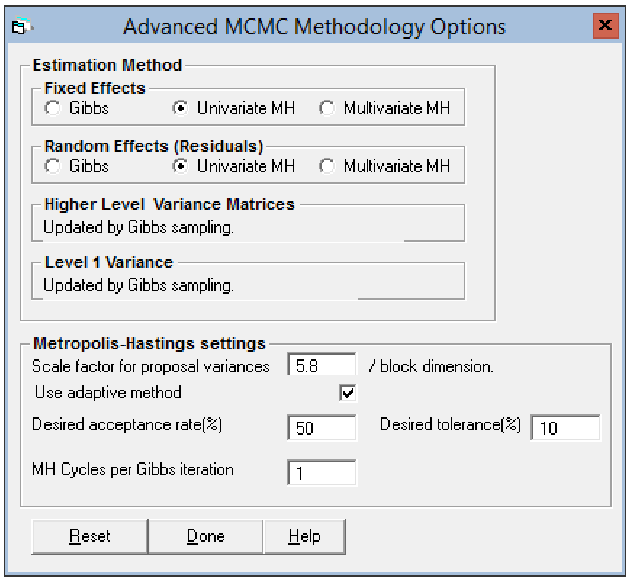
25.6 Normal response models in MLwiN

We alluded to the fact earlier that for Gibbs sampling in MLwiN we have not programmed up the full hierarchical centring algorithm. As was mentioned in section 25.3 when used with non-Normal responses, as MLwiN uses random walk Metropolis sampling for both level 2 residuals and fixed effects, the only change to the algorithm is a new Gibbs step for fixed effects that are centred and an adjustment to the residuals that results in correct conditioning in the step.

This can also be used in MLwiN with Normal responses and random walk Metropolis sampling as detailed below. Firstly return to the beginning of this chapter for instructions on setting up the model for the **tutorial** dataset and setting up hierarchical centring prior to generating WinBUGS code. We then need to change estimation methods used:

- Select **MCMC/MCMC methods** from the **Model** menu.
- Click on the **Univariate MH** buttons for both **Fixed Effects** and **Random effects**.

The **Advanced MCMC Methodology** window should look as follows:


Advanced MCMC Methodology Options

Estimation Method

Fixed Effects: Gibbs [✓ Univariate MH] Multivariate MH

Random Effects (Residuals): Gibbs [✓ Univariate MH] Multivariate MH

Higher Level Variance Matrices: Updated by Gibbs sampling.

Level 1 Variance: Updated by Gibbs sampling.

Metropolis-Hastings settings

Scale factor for proposal variances: 5.8 / block dimension.

Use adaptive method: ✓

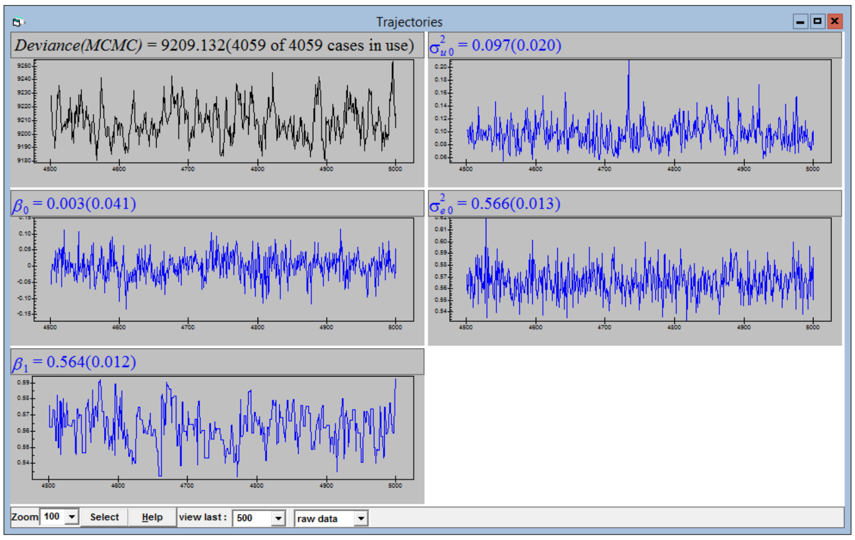
Desired acceptance rate(%): 50 Desired tolerance(%): 10

MH Cycles per Gibbs iteration: 1

- Click on the **Done** button on the **Advanced MCMC Methodology** window

- Click on the **Start** button

After running for 5,000 iterations the **Trajectories** window will look as follows:


Trajectories
Deviance(MCMC) = 9209.132(4059 of 4059 cases in use)
σ^2_{u0} = 0.097(0.020)
β_0 = 0.003(0.041)
σ^2_{e0} = 0.566(0.013)
β_1 = 0.564(0.012)
Zoom 100
Select
Help
view last : 500
raw data

It is easy here to pick out that β_0 which is pushed up the hierarchy has been updated using Gibbs sampling whilst β_1 is using Metropolis Sampling. If we compare the ESS for all parameters with those from WinBUGS earlier we see the results given in the 3rd and 4th columns below:

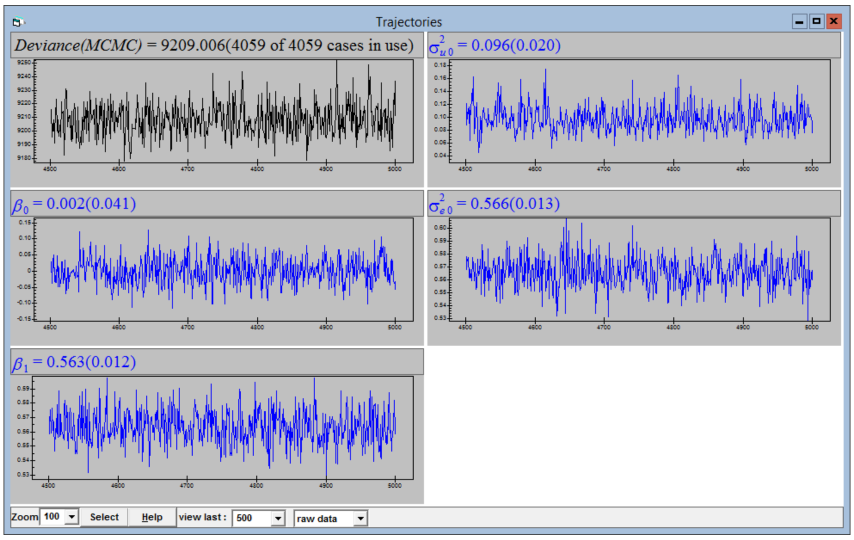
Parameter	Non Centred	WinBUGS	MH	Gibbs
β_0	216	3754	2750	4959
β_1	4413	4206	905	4212
σ^2_{u0}	2821	2874	2149	3064
σ^2_{e0}	4712	4815	4145	5179

The final option we have is to use a 'pseudo' hierarchical centring algorithm with Gibbs sampling. Here we give a caution that the algorithm we use has not been confirmed as a correct MCMC algorithm but does appear to give sensible estimates and good ESS. Basically we now need to rerun the model above using Gibbs sampling. As we have previously used Metropolis we need to change back to Gibbs Sampling. To do this we do the following:

- Click on the **IGLS/RIGLS** tab on the **Estimation control** window.
- Click **Start**.
- Click on the **Estimation Control** button.
- Select the tab labelled **MCMC** on the **Estimation control** window.

- Select **MCMC/MCMC methods** from the **Model** menu.
- Click on the **Reset** button and then the **Done** button on the **Advanced MCMC Methodology** window.
- Click on the **Start** button.

After running we get the following chains in the **Trajectories** window:



Deviance(MCMC) = 9209.006(4059 of 4059 cases in use)

σ^2_{u0} = 0.096(0.020)

β_0 = 0.002(0.041)

σ^2_{e0} = 0.566(0.013)

β_1 = 0.563(0.012)

Zoom 100 | view last : 500 | raw data

Here we see all parameters appear to mix well and be using Gibbs sampling steps. The ESS values are given in the last column of the earlier table and are as good if not better than those for the true hierarchical centring algorithm used by WinBUGS. So the question is what algorithm is MLwiN using?

If we consider the standard (uncentred) Gibbs sampling algorithm we have four sets of steps to update β, u, σ^2_u and σ^2_e respectively. We can also split β into two parts, β^C which can be centred and β^U which can't. Now the Metropolis sampling algorithm used earlier would have updated β^U using univariate updating steps and simply replaced the equivalent univariate steps for β^C with a Gibbs step with an adjustment to u_j so that in reality $u^*_j = u_j - (X^C\beta^C)_j$, the parameter from a centred formulation, is held constant and hence conditioned on.

So what we have done for the Gibbs algorithm is supplemented the four steps for the non-centred formulation with the Gibbs step for β^C conditioning on the u^*_j. This means that in each iteration β^C is updated in two steps and u_j is updated in one step and adjusted in another. This may explain the better mixing even than the standard Gibbs algorithm we demonstrated using WinBUGS. Of course it is not clear that this algorithm is an acceptable Gibbs sampling algorithm and we hope any MCMC theory researchers out there can let us know either way. Until we have confirmation we therefore urge caution while using this option and hence have left it to the end of the chapter.

Chapter learning outcomes

- ⋆ How and when to use hierarchical centring to improve mixing.
- ⋆ How to use hierarchical centring and the WinBUGS interface.
- ⋆ Hierarchical centring for non-Normal response models.
- ⋆ The use of a psuedo hierarchical centring algorithm in MLwiN for Normal responses.

Bibliography

Albert, J.H. & Chib, S. (1993). Bayesian analysis of Binary and Polychotomous Response Data. *Journal of the American Statistical Association*, **88**:669–679.

Besag, J., York, J. & Mollie, A. (1992). Bayesian image restoration with two applications in spacial statistics (with discussion). *Annals of the Institute of Statistical Mathematics*, **43**:1–59.

Best, N., Cowles, M.K. & Vines, K. (1995). Coda: Convergence diagnosis and output analysis software for Gibbs sampling output, Version 0.3. Technical report, Medical Research Council Biostatistics Unit, Cambridge.

Breslow, N.E. & Clayton, D.G. (1993). Approximate inference in generalized linear mixed models. *Journal of the American Statistical Association*, **88**:9–25.

Browne, W.J. (1998). *Applying MCMC methods to multilevel models*. Ph.D. thesis, University of Bath.

Browne, W.J. (2004). An illustration of the use of reparameterisation methods for improving MCMC efficiency in crossed random effects models. *Multilevel Modelling Newsletter*, **16**(1):13–25.

Browne, W.J. (2006). MCMC algorithms for constrained variance matrices. *Computational Statistics & Data Analysis*, **50**(7):1655–1677.

Browne, W.J. & Draper, D. (2000). Implementation and performance issues in the Bayesian and likelihood fitting of multilevel models. *Computational Statistics*, **15**:391–420.

Browne, W.J. & Draper, D. (2006). A comparison of Bayesian and likelihood-based methods for fitting multilevel models. *Bayesian Analysis*, **1**:473–550.

Browne, W.J., Goldstein, H. & Rasbash, J. (2001*a*). Multiple Membership Multiple Classification (MMMC) models. *Statistical Modelling*, **1**:103–124.

Browne, W.J., Goldstein, H., Woodhouse, G. & Yang, M. (2001*b*). An MCMC algorithm for adjusting for errors in variables in random slopes models. *Multilevel Modelling Newsletter*, **13**(1):4–10.

Browne, W.J., Akkol, S. & Goldstein, H. (2009*a*). MCMC algorithms for structured multivariate normal models. Technical report, submitted for publication.

Browne, W.J., Steele, F., Golalizadeh, M. & Green, M. (2009*b*). The use of simple reparameterisations to improve the efficiency of MCMC estimation for multilevel models with applications to discrete-time survival models. *Journal of the Royal Statistical Society, Series A*, **172**:1–2.

Browne, W.J., Draper, D., Goldstein, H. & Rasbash, J. (2002). Bayesian and likelihood methods for fitting multilevel models with complex level-1 variation. *Computational Statistics and Data Analysis*, **39**(2):203–225.

Chatfield, C. & Collins, A.J. (1980). *Introduction to Multivariate Analysis.* London: Chapman & Hall.

Chib, S. & Carlin, B.P. (1999). On MCMC sampling in hierarchical longitudinal models. *Statistics and Computing*, **9**:17–26.

Chib, S. & Hamilton, B.H. (2000). Bayesian analysis of cross-section and clustered data selection models. *Journal of Econometrics*, **97**(1):25–50.

Clayton, D.G. & Kaldor, J. (1987). Empirical Bayes estimates of age-standardized relative risks for use in disease mapping. *Biometrics*, **43**:671–681.

Elliott, P., Wakefield, J.C., Best, N.G. & Briggs, D.J. (2000). *Spatial Epidemiology: Methods and Applications.* Oxford: Oxford University Press.

Gamerman, D. (1997). Sampling from the posterior distribution in generalized linear mixed models. *Statistics and Computing*, **7**:57–68.

Gelfand, A.E., Hills, S.E., Racine-Poon, A. & Smith, A.F.M. (1990). Illustration of Bayesian inference in Normal data models using Gibbs Sampling. *Journal of the American Statistical Association*, **85**:972–985.

Gelfand, A.E., Sahu, S.K. & Carlin, B.P. (1995). Efficient parameterisations for normal linear mixed models. *Biometrika*, **82**:479–488.

Gelman, A. (2006). Prior distributions for variance parameters in hierarchical models. *Bayesian Analysis*, **1**:515–533.

Gelman, A., Roberts, G.O. & Gilks, W.R. (1995). Efficient Metropolis jumping rules. In J.M. Bernado, J.O. Berfer, A.P. Dawid & A.F.M. Smith, eds., *Bayesian Statistics 5*, pages 559–607. Oxford: Oxford University Press.

Gelman, A., van Dyk, D.A., Huang, Z.Y. & Boscardin, W.J. (2008). Using redundant parameterizations to fit hierarchical models. *Journal of Computational and Graphical Statistics*, **17**:95–122.

Geweke, J. & Zhou, G. (1996). Measuring the pricing of the arbitrage pricing theory. *The Review of Financial Studies*, **9**:557–587.

Gilks, W.R. & Wild, P. (1992). Adaptive rejection sampling for Gibbs sampling. *Journal of the Royal Statistical Society, Series C*, **41**:337–348.

Gilks, W.R., Richardson, S. & Spiegelhalter, D.J. (1996). *Markov Chain Monte Carlo in practice*. London: Chapman & Hall.

Goldstein, H. (2003). *Multilevel statistical models*. London: Arnold, 3rd edition.

Goldstein, H. & Browne, W.J. (2002). Multilevel factor analysis modelling using Markov Chain Monte Carlo (MCMC) estimation. In Marcoulides, G. A. & Moustaki, I., eds., *Latent Variable and Latent Structure Models*, pages 225–243. New Jersey: Lawrence Erlbaum.

Goldstein, H. & Speigelhalter, D.J. (1996). League tables and their limitations: statistical issues in comparisons of institutional performance. *Journal of the Royal Statistical Society, Series A*, **159**:385–443.

Kass, R.E., Carlin, B.P., Gelman, A. & Neal, R. (1998). Markov Chain Monte Carlo in practice: a roundtable discussion. *American Statistican*, **52**:93–100.

Knorr-Held, L. & Rue, H. (2002). On block updating in Markov random field models for disease mapping. *Scandinavian Journal of Statistics*, **29**:597–614.

Krzanowski, W.J. & Marriott, F.H.C. (1994). *Multivariate Analysis: Part 1: Distributions, Ordination and Inference*. London: Edward Arnold.

Krzanowski, W.J. & Marriott, F.H.C. (1995). *Multivariate Analysis: Part 2: Classification, Covariance Structures and Repeated Measurements*. London: Edward Arnold.

Langford, I.H., Bentham, G. & 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.

Langford, I.H., Leyland, A.H., Rasbash, J., Goldstein, H., Day, R.J. & Macdonald, A.L. (1999). Multilevel modelling of area-based health data. In A.B. Lawson, A. Biggeri, D. Boehning, E. Lessafre, J. Viel & R. Bertollini, eds., *Disease Mapping and Risk Assesment for Public Health*. Chichester: Wiley.

Lawson, A.B., Biggeri, A., Boehning, D., Lessafre, E., Viel, J. & Bertollini, R. (1999). *Disease Mapping and Risk Assesment for Public Health*. Chichester: Wiley.

Lawson, A.B., Browne, W.J. & Roderio, C.L.V. (2003). *Disease Mapping with WinBUGS and MLwiN*. Chichester: Wiley.

Liu, C., Rubin, D.B. & Wu, Y.N. (1998). Parameter expansion to accelerate EM: The PX-EM algorithm. *Biometrika*, **85**:755–770.

Liu, J.S. & Wu, Y.N. (1999). Parameter expansion for data augmentation. *Journal Of The American Statistical Association*, **94**:1264–1274.

McCulloch, C.E. & Searle, S.R. (2001). *Generalized, Linear and Mixed Models.* New York: Wiley.

Mollie, A. (1996). Bayesian mapping of disease. In W.R. Gilks, S. Richardson & D.J. Spiegelhalter, eds., *Markov Chain Monte Carlo in Practice.* London: Chapman and Hall.

Mortimore, P., Sammons, P., Stoll, L., Lewis, D. & Ecob, R. (1988). *School Matters.* Wells: Open Books.

Raftery, A.E. & Lewis, S.M. (1992). How many iterations in the Gibbs sampler? In J.M. Bernado, J.O. Berfer, A.P. Dawid & A.F.M. Smith, eds., *Bayesian Statistics 4*, pages 765–76. Oxford: Oxford University Press.

Rasbash, J. & Browne, W.J. (2001). Modelling non-hierarchical structures. In A.H. Leyland & H. Goldstein, eds., *Multilevel Modelling of Health Statistics*, pages 93–105. Chichester: Wiley.

Rasbash, J. & Browne, W.J. (2002). Non-hierarchical multilevel models. In J.D. Leeuw & E. Meijer, eds., *Handbook of Multilevel Analysis.* Springer.

Rasbash, J. & 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.

Rasbash, J., Goldstein, H., Browne, W.J., Yang, M. & Woodhouse, G. (2000). *The MLwiN Command Interface version 1.3.* Institute of Education, University of London, London.

Rasbash, J., Steele, F., Browne, W.J. & Goldstein, H. (2008). *A User's Guide to MLwiN Version 2.10.* Centre for Multilevel Modelling, University of Bristol, Bristol.

Rubin, D.B. (1976). Inference and missing data. *Biometrika*, **63**:581–592.

Rubin, D.B. (1987). *Multiple Imputation for Nonresponse in Surveys.* New York: J. Wiley and Sons.

Rue, H. (2001). Fast sampling of Gaussian Markov Fields. *Journal of the Royal Statistical Society, Series B*, **63**:325–338.

Sargent, D.J., Hodges, J.S. & Carlin, B.P. (2000). Structured Markov chain Monte Carlo. *Journal Of Computational And Graphical Statistics*, **9**:217–234.

Schafer, J.L. (1996). *Analysis of Incomplete Multivariate Data.* London: Chapman & Hall.

Schafer, J.L. (1997). Imputation of missing covariates under a multivariate linear mixed model. Technical report, unpublished.

Seltzer, M.H. (1993). Sensitivity analysis for fixed effects in the hierarchical model: A Gibbs sampling approach. *Journal of Educational Statistics*, **18**:207–235.

Spiegelhalter, D.J., Thomas, A. & Best, N.G. (2000*a*). *WinBUGS version 1.3: user manual.* Medical Research Council Biostatistics Unit, Cambridge.

Spiegelhalter, D.J., Thomas, A. & Best, N.G. (2000*b*). *WinBUGS version 1.3: Examples volume II.* Medical Research Council Biostatistics Unit, Cambridge.

Spiegelhalter, D.J., Best, N.G., Carlin, B.P. & van der Linde, A. (2002). Bayesian measures of model complexity and fit (with discussion). *Journal of the Royal Statistical Society, Series B*, **64**:191–232.

Tanner, M. & Wong, W. (1987). The calculation of posterior distributions by data augmentation (with discussion). *Journal of the American Statistical Association*, **82**:528–550.

Thomas, A., Best, N.G. & Spiegelhalter, D.J. (2000). *GeoBUGS User Manual, Demonstration Version 1.0.* Imperial College and MRC Biostatistics Unit, Cambridge.

van Dyk, D.A. & Meng, X.L. (2001). The art of data augmentation. *Journal of Computational and Graphical Statistics*, **10**:1–50.

Woodhouse, G., Yang, M., Goldstein, H. & Rasbash, J. (1996). Adjusting for measurement errors in multilevel analysis. *Journal of the Royal Statistical Society, Series A*, **159**:201–212.

Yang, M. & Woodhouse, G. (2001). Progress from GCSE to A and AS level: Simple measures and complex relationships. *British Educational Research Journal*, **27**:245–268.