Example session MLwiN

*Tom A.B. Snijders**

March 2007

Contents

*University of Oxford and University of Groningen ,
http://stat.gamma.rug.nl/snijders/ .

Introduction

This is an introduction to MLwiN version 2.0. The introduction is built up by guiding you through a MLwiN session, which uses data from Snijders & Bosker (1999). It is assumed that you know the basics of working with Windows (clicking, dealing with menus and windows, etc.) For more information about MLwiN, you may consult the extensive user's manual, which can be downloaded from
`http://www.mlwin.com/download/index.html` ;
but the Help facility within MLwiN may make this written documentation almost superfluous.

Sections 1-3 treat the basic operations of MLwiN. For those using this as a tutorial in MLwiN, it is advised that they study these sections in any case, and the following sections depending on their interests and research needs.

1 The start

First start MLwiN. Below the title bar you see the menu bar:

File Edit Options Model Estimation Data Manipulation Basic Statistics Graphs Window Help

First have a look at what is behind these menu items by clicking on each of them and briefly looking at the list of commands available there.

In the File menu, there are various possibilities that will be needed to start working. The "worksheet" mentioned there is the name for the MLwiN "system file". The first time you start working with MLwiN on a new data set, you will have to input the data from an ASCII text file (a raw text file) or import it from another program using the normal Windows cut-and-paste method. During or at the end of the session you can save the worksheet, which you can then open later when resuming work on this dataset. The option Print Window Image can be used to print the window you are currently working in to help you remember the past.

Have an extensive look in the Help menu so that you get an idea about how you will be able to get help when actually working with MLwiN. In any case have a look at the following:

1. In the Contents, go to Introduction to MLwiN and on to Introduction to MLwiN. Have a look at this text, and continue by clicking the >> button in the top menu bar for this help screen. It will be useful to have a glance at the 16 sections up to and including the page titled 'Missing data' to know the kind of explanation you can get from this help system.

2. Also go to the Index. There you find a very long list of keywords. E.g., look at each of the keywords Multilevel data structures, then Multilevel models, and finally Multilevel data manipulations.

3. For an impression of the way in which this Help system gives more specific information, look at Calculate window and at FAQ.

The main menus for working with MLwiN are Model, which is used for defining, estimating, and examining various models; and Data Manipulation, which

is used for viewing data, recoding, defining new variables (e.g. aggregates over groups), etc.

1.1 Commands

You can tell MLwiN what to do in two ways: by clicking (the 'true Windows' procedure) and by giving written *commands*. Most of this introduction is about the use of clicking. But sometimes commands come in handy: they can give enhanced flexibility and they are convenient for repetitive tasks. Commands can be collected in *macros*. A macro is a list of commands to be executed consecutively, which are collected in an ASCII file, similar to e.g. a syntax file in SPSS.

Commands can be given by typing them in the bottom line of the Command Interface[1] in the Data Manipulation menu, and then hitting the Return key.

Macros can be written and executed in the Macro Editor which can be accessed through the File menu by choosing New Macro or Open Macro. (Macros can also be executed by using the Obey command in the Command Interface. The Macro Editor gives more flexibility than the Command Interface, and it is advisable to use the Macro Editor.)

1.2 Preparations for working with MLwiN

Depending on the installation of MLwiN, it will usually be necessary to select the "current directory" in which MLwiN will look for data files, where it will store results, etc. To do this, click on Options – Directories. Now you are in the Directories tab; change the Current Directory to the desired directory. (If you are working on MLwiN installed in a network, the default directory here may well be a network drive for which you have no write permission, and forgetting to change it may lead to error messages or even an abortive end of the MLwiN session.) Leave the rest at the default settings.

There are several ways to record what you have done in a MLwiN session. You can note on a piece of paper everything that is relevant on the screen. An alternative is to use the well-known "Copy" facility, which you find under the Edit menu item but for which you can use also the <Ctrl>-C key. Clicking Copy or typing <Ctrl>-C will copy the contents of the current window (or the selected part, if you selected a part with the mouse) to the clipboard. You can then go to a word processor such as *MS-Word* and paste these clipboard contents to the file being edited there. (You may have to use a "Paste special" command in such a program.) A third possibility is to use the Print Window Image item in the File menu. A fourth way is provided by the *log file*. This log file records the output of commands issued in the command interface or in macros; or, hidden to you, by the program.[2]

To request a log file choose Data Manipulation – Command Interface. Give a mouse click in the bottom line of the Command Interface. There type in

```
LOGON EXA1.LOG
```

[1] The downloadable MLwiN manuals comprise a special manual for the Command interface.

[2] The program does not record estimation results unless you request this by the commands fixed, random, and like.

where you may change the file name (*exa1.log*) to any file name you prefer, and press the Return key. (If the program tells you that it cannot carry out this command, the reason is probably that it assumes a default directory on the network, for which you have no permission to write a file. In this case, you can either give the file name explicitly with the full drive and path or select another current directory in the Options menu, as indicated above.) The LOGON command[3] makes MLwiN save a history of your session in the log file. What is in the output window also is stored in the log file.

1.3 Data input

To understand the operation of MLwiN, you should know that MLwiN stores data (variables) internally in "columns" (or vectors), indicated by *C1, C2*, etc., up to *C1500* (the number of 1500 is a default, which can be altered in the Options – Worksheet menu). In addition, constants can be stored in "boxes", indicated by *B1* to *B400*. Columns *C1090* through *C1099* are reserved for model calculations and should not be used to store data.[4] To see what variables you have in your data set, choose Data Manipulation – Names and you get the list of the 1500 columns with the variable names and for each column, or variable, also n, the number of cases.

There are four basic ways to get data into MLwiN:

1. direct input of ASCII ("raw") data, in free or fixed format;
2. cut-and-paste from data stored in another Windows program on your PC;
3. input of ASCII data by means of a macro;
4. input of a worksheet (the MLwiN name for its system file).

For SPSS users, the `PreML` facility described at the end of Section 1.3.2 may be very useful.

For the estimation of multilevel models, it is essential that the data are ordered according to the multilevel structure – i.e., groups in the multilevel structure should be connected parts in the data set without being interrupted by data for other groups. If this does not hold for the input data set, it can be achieved later by sorting the data, but it will be easy to use an original data set which is already ordered correctly.

MLwiN assumes in many of its operations that data are complete, except for data that have the value of the missing data code (e.g., 999) defined in the Options – Numbers menu item. If you have data which includes some missings and you import them from another program into MLwiN, then usually it is best to represent the missings in the original program by some code such as 999, and then export to MLwiN while using this code as a data value. MLwiN uses only one value to represent missings.

[3] LOGON will overwrite a possibly earlier existing file with the same name. I you do not wish this, use LOGA – for LOGAppend – to append the new log at the end of the existing file with this name.

[4] If you use bootstrap or quasilikelihood procedures, some more columns are reserved; see Columns reserved in the Help Index.

1.3.1 Direct input of ASCII data

In this session another way to input data will be used in practice, as explained in the next subsection; but to see how MLwiN works, first we describe how a raw data file can be read. (When you follow this as a practical example session, you can just read this subsection and keep it for later reference, if ever.)

In free format data, the variables in the data set must be separated by blanks, and missings are not allowed (although you may use numerical values such as 999 as a code for missing data; the missing value code is defined in the Options – Numbers menu item). Click on the File menu and then choose ASCII text file input. You can choose Browse to look for the file anywhere on your disk, but you can also select the directory-plus-filename if you know the whereabouts of the file. For columns, you must know how many variables there are in the data set, and decide on the columns in which you would like to store them. E.g., if the data set contains twenty-five variables, you could store them in columns *C1-C25* but also in *C40-C64*; or *C1, C5, C9, C21-C42*. Indicate this by typing the desired columns, e.g., *C1-C25*, in the Columns field. Click OK and the data file will be input.

If the variables in the data file are not always separated by blanks, or if missing values are indicated by blanks, it is necessary to use formatted data input. You then have to tell the program the positions of the variables, i.e., the "format" of the file. If you need this option, click the ? Help button within the ASCII text file input window for further explanation.

For *cut-and-paste* data input from another program, consult the cut keyword in the Help Index. An advantage of this method over raw ASCII data input, is that you can take along the variable names if these are available in the program from which the data are taken.

1.3.2 Input of ASCII data via a macro

It can be convenient to combine raw data with variable names (and perhaps comments, variable transformations, initial model definition, etc.) in one file that can be submitted to MLwiN as a macro via the Command Interface. This will be illustrated here in an example.

We shall go through some examples that are also in Chapters 4 and 5 of the textbook, *Multilevel Analysis: An introduction to basic and advanced multilevel modeling*, by Tom A.B. Snijders and Roel J. Bosker (Sage, 1999). The dataset macro is called MLBOOK1.DAT and it is included in the zipped file collection MLBOOK.ZIP which can be downloaded from
`http://stat.gamma.rug.nl/snijders/mlbook1.htm`.

This macro is an ASCII (text) file. Have a look at the file by means of some text editor (be careful not to save it in Word or Wordperfect format, or as any kind of file other than ASCII or Text, if you want it to be read by MLwiN). This file has in the beginning and at the end a number of MLwiN commands, and a large data set in between. You see that the file begins as follows (the lines are longer and have been truncated for this display):

```
echo 0
assign c1-c25
1.000 1.000 15.00 12.33 0.000 0.000 0.000 14.00 180.0 24.00 36.00 46.00 .....
1.000 2.000 14.50 10.00 0.000 1.000 0.000 12.00 180.0 19.00 36.00 45.00 .....
```

```
1.000 3.000 9.500 11.00 0.000 0.000 0.000 10.00 180.0 24.00 33.00 33.00 .....
1.000 4.000 11.00 10.00 0.000 0.000 0.000 13.00 180.0 26.00 29.00 46.00 .....
1.000 5.000 8.000 6.666 0.000 0.000 0.000 8.000 180.0 9.000 19.00 20.00 .....
1.000 6.000 9.500 9.000 0.000 1.000 0.000 8.000 180.0 13.00 22.00 30.00 .....
```

The first line tells MLwiN not to echo each data line read to the output window. The second line announces MLwiN that there will follow data input for the columns *C1* to *C25*. What is then reproduced here are two series of 25 data values. MLwiN understands by itself that it has to continue with the next line, so it is not important how many lines are used for each case.

Looking at the end of the file shows that the data ends as follows:

```
258.0 2283. 12.50 5.333 1.000 0.000 0.000 10.00 25880 11.00 24.00 21.00 ......
258.0 2284. 9.000 8.666 1.000 0.000 0.000 9.000 25880 6.000 22.00 33.00 ......
258.0 2285. 11.00 12.33 0.000 0.000 1.000 13.00 25880 14.00 25.00 26.00 ......
258.0 2286. 10.50 9.333 1.000 0.000 0.000 9.000 25880 11.00 31.00 34.00 ......
258.0 2287. 12.00 10.33 1.000 0.000 0.000 8.000 25880 8.000 29.00 39.00 ......

finish
echo 1
name c1 "schoolNR"
name c2 "pupilNR"
..............
name c23 "homework"
name c24 "classsiz"
name c25 "groupsiz"
```

The command finish comes after the last data line, then the command that what follows is indeed to be echoed to the output window or log file, then follow the commands that give the names for the 25 variables. Including these names makes such a macro very convenient for data input. If you wish to include comments, they can be given by starting a line (but not in the middle of the data lines) by the command note, which tells MLwiN to skip this line.

To use this form of data input for reading the variables contained in this file, which is based on a macro, you must employ one of the two ways of executing a macro: by the Macro Editor in the File menu, or by the Command Interface, which is accessible from the Data Manipulation menu.

We use the second way, because due to the large length of the file, the Macro Editor is very slow to read it. Choose Data Manipulation – Command Interface. This will also open the Output window (you can close and open this by pressing the Output button in the upper left corner of the Command Interface). Give a mouse click in the bottom line of the Command Interface In this bottom line type in

```
OBEY MLBOOK1.DAT
```

(Depending on the version of MLwiN you may get a warning screen which basically says that the file contains numbers with a lot of decimals, and numbers will be rounded to 6 significant digits. This is not a problem, therefore just click on the Done button on this warning screen and the program will continue.)
After any macro is executed, the output window contains both the commands in the macro and the responses of MLwiN – if any – to the commands; but the macro commands between the `echo 0` and `echo 1` statements are omitted. Note from the Name commands in the macro, that the first variable (*C1*) is the school number and the second variable (*C2*) the pupil number.

To see what you have done, click Data Manipulation – Names. You see the 25 defined variables, each with 2287 cases, and their minimum and maximum values. This helps to check whether data definition and input proceeded correctly. Now select Data Manipulation – View or edit data. You will see the data in some of the variables for some of the cases. To see other variables, click on the "View" field and select the desired variables. By holding down the <Ctrl> button while you are clicking, you can increase the number of variables to be viewed. If you click on variables *C25* and *C26*, you will see that *C25* contains data and *C26* is empty. In the top of the data window you see the number of cases for each column; it is 2287 for this data set. At this moment, all columns still have the same number of cases, but after some operations you may have created columns with varying numbers of cases. The number of cases in a column is also called by MLwiN the "length" of this column. To change variables names or give names to newly created columns, you can use the Names window. In this window, you can click on a variable, then move the mouse to the window above the list of variables, enter the desired name, and confirm this name by pressing the Return key. For 25 variables, this is a lot of work, which is the reason why the names were included in the MLBOOK1.DAT macro file.

The other way to execute macros is by using the Macro Editor, which is accessible through the File menu. In this menu, choose Open Macro and choose the file MLBOOK1.DAT. A problem is, however, that this is a very long file, and a pretty long time will be required by the macro editor for reading it. Therefore this method is not chosen in this session.

For SPSS users it can be convenient to use the SPSS include file `PreML.inc`, written by Jurjen Iedema and available as part of the set of MLwiN macros on `http://stat.gamma.rug.nl/snijders/multilevel/`. This include file allows you to select variables and indicate the grouping variable for the multilevel structure, and makes SPSS write the data set plus the further information to a macro for use in MLwiN.

1.4 Some data transformations

The first thing to do now is to make available some transformed variables. Transforming variables can be done by various of the items in the Data Manipulation – Calculate window; or through commands/macros.

1.4.1 The constant

A peculiarity of MLwiN is that it requires you to make a "constant variable", with all values equal to 1, which usually is called *cons* (but that is up to you). An extra requirement is that you must tell MLwiN that this variable must have as many cases as the data set you wish to analyze, in this case 2287 (in MLwiN terminology, this column must have length 2287). There are various ways to do this. Two ways are explained here.

One way is to choose Data Manipulation – Calculate; in the blank space in the upper right part of this window, type in

```
c26 = 1 + 0*c1
```

and press the Calculate button in the bottom of this Calculate window. Since column *C1* has length 2287, the new column *C26* will have this length, too.

(In this window you can click the **?** Help button to be instructed about the possibilities for calculations.) Now go to Data Manipulation – Names, select the new variable *C26*, click on the blank space in the upper left corner of this window, type

```
cons
```

and press the *Enter* key.

Some alternative ways to do the same thing are explained here, in order to give you some understanding of the use of commands and the Macro Editor. In the File menu, click on New Macro. A new macro will be opened, as yet with the name Untitled;1. Take care that the Output window (connected to the Command Interface) is open. In the editing field of the Macro Editor, type the two lines

```
code 1 1 2287 c26
name c26 "cons"
```

The code command defines a new variable *C26* with all 2287 values equal to 1, which subsequently gets the name *cons*. Click on Execute in the bottom line of the Macro Editor. Then you will see these two commands echoed in the Output window. In the Names window, you can check that indeed the new variable has been created.

Instead of the code command, you could also use the calc command, which follows the same syntax as what is used in the Data Manipulation – Calculate window, and in this case means that instead of the `code 1 1 2287 c26` line you use the command

```
calc c26 = 1 + 0*c1
```

The descriptions of these commands can be found in the Help window by going to the index and looking for command code and command name.

By typing in these lines, you just made a (very short) MLwiN macro. For convenience, it is advisable to save the macro. This is done by going to the File menu and click on Save Macro. You will then be prompted for the name and location of the macro. It will be most convenient to store it in the same directory where also the MLBOOK1.DAT file is stored.

1.4.2 Dummy variables

The variable *C14* contains the schools' denominations. To transform this into dummy variables, use the command DUMMY. Since MLwiN only cares about the first four letters of any command name, you can type DUMM or DUMMBUTFUN instead of DUMMY if you really want to. In the Macro Editor, type

```
dumm c14 c27-c29
name c27 "catholic"
name c28 "protestant"
name c29 "nondepri"
```

(see Example 6.3 (p. 89) in Snijders & Bosker (1999); the reference category *C14* = 1 is public (non-denominational) schools; '*nondepri*' stands for *non-denominational private* schools). The DUMMy command expects that the values

of the first mentioned (input) variable (here *C14*) are 1,2, ..., k for some number k; next to this input column, k or $k-1$ output columns must be mentioned. Use the View or Edit Data window to see how variables *C14, C27, C28*, and *C29* hang together.

When you now click Execute, the whole macro will be executed, including the first two lines for the creation of *cons*. It is not a problem to repeat this creation. Save the macro again.

1.4.3 First multilevel data manipulations

Before further data transformation, suppose we want to get the distribution of group sizes as information about the data set, possibly to be used in later calculations. Go to the window Data Manipulation – MultiLevel Data Manipulations, choose Operation: Count , On blocks defined by: schoolnr and select Free columns. Column *C30* will then appear as Output Column, because this is the first free column. Now click on Add to action list and then on Execute. In the Names or View data window, you can see that this has created a new variable of, again, 2287 cases; for each pupil it gives the number of pupils in his or her school in this data set.

It is instructive to do this while having the Command Interface open, and after having unchecked the User field in the Command Interface. (This field indicates whether you wish to see in the Command Interface only those commands that you have typed in as a user, or also the commands that corresponds to clicking operations.) Then, upon executing the data manipulation, a few commands appear in the Command Interface, of which the crucial one is

```
MLCOunt "schoolnr" C30
```

This teaches you that MLCO, abbreviating MLCOUNT, is the command for writing the counts corresponding to the level indicator schoolnr to the new variable *C30*. To know more about this command, look at Command MLCO in the Help index.

If we wish to have a new variable that includes each school size only once per school, instead of being replicated for each pupil in the school, go to the Data Manipulation – unreplicate window, choose Input column *C30*, Take first entry in blocks defined by schoolnr, and select Free columns. Again click on Add to action list and then on Execute. This creates a variable *C31* of 131 cases, one case per school. This unreplicate or Take data facility is designed to transform higher-level variables, present in the data set in replicated form, to a variable where each unit at his level (here: each school) is represented by only one data case.[5]

If you wish to analyze variables at the school level, then the same procedure can be followed for other school-level variables. E.g., the Input column mixedgra could be used and treated in the same way. This variable, called *COMB* in Snijders & Bosker (1999, p. 76), indicates classes that are a combination of grade 7 and grade 8 pupils; the current data set contains only the grade 8 pupils. Now the correlation (on the school level) between group size and mixed grade classes

[5]Looking at the Output window as before, with unchecked user box in the Command Interface, shows that unreplicating can be carried out by the command `TAKE "schoolnr" c30 C1500 C31`. This will also create a superfluous column *C1500*, which subsequently is erased again by the MLwiN macro system.

can be calculated in the Basic Statistics – Averages and Correlations window; the correlation is shown in the Output window of the Command Interface and is equal to −0.66, confirming that small grade groups were pooled together.

1.4.4 Global centering

The next steps are to subtract the global averages from the variables *C3* (named *iq_verb*, verbal intelligence), *C13* (*ses*), and *C25* (*groupsize*). First calculate these averages by going to the Basic Statistics – Averages and Correlations window, selecting these three variables (click on them while keeping the <Ctrl> key pressed so that the earlier variable selection is retained), and requesting the averages. The results are 11.834, 27.812, and 23.101.

Now go to Data Manipulation – Calculate and there type in

```
c3 = c3 - 11.834
```

and after that press the *Enter* key or press the Calculate button (not both, because then you would have subtracted this amount twice). Similarly calculate

```
c13 = c13 - 27.812
```

and

```
c25 = c25 - 23.101
```

Go to the Names window to check the minimum and maximum of these variables – since they should now have an average of 0, the minimum should be negative and the maximum positive. If they don't, you made an error.

Instead of using Data Manipulation – Calculate, you can also type

```
calc c3 = c3 - 11.834
calc c13 = c13 - 27.812
calc c25 = c25 - 23.101
```

in the Macro Editor. An example of always using the Macro Editor for these purposes is that you know exactly what you did, and you can easily repeat your actions.

1.4.5 Save the worksheet

At this moment, it is wise to save the worksheet (remember, in the File menu). Otherwise the results of your efforts would be lost in case of a failure of the power, the computer, the program, or perhaps yourself. You will be asked to give a name; e.g., you can choose the name *exa1.ws*. The extension *ws* is a convenient extension for worksheets. If you get an error message when trying to save the worksheet, this probably means that you are (unwittingly) trying to write on a network drive for which you have no write permission. In that case, either give the full path name or choose another current directory.

2 Model definition

Go back to the View Data window in order to have a look at the data again. Look at the first two variables, *pupilnr* (*C2*) and *schoolnr* (*C1*). You see that the 2287 cases are ordered as follows: the schools define the main order and within the schools the cases are ordered by pupil number. Such an ordering is required by the algorithm of MLwiN; if the data are not ordered correctly, use Data Manipulation – Sort.

The first thing to do after data input is the definition of the nesting structure and of the dependent variable. The model definition is most conveniently given and examined in the Model – Equations window. In this window, click on the y that is in the upper left corner of this equations window. Now the Y variable window pops up. The "Y variable" is how MLwiN refers to the dependent variable, also called the response variable. Choose *langpost* (the posttest on language) as the dependent variable and choose 2 as the number of levels. Now you can choose the variables that identify the levels: choose *schoolnr* as the identifier for level 2 and *pupilnr* as the identifier for level 1. For a valid nesting structure, it is required that the cases belonging to one level-2 unit are contiguous and that they are ordered within the level-2 unit by the identifying numbers of the level-1 units. It is not a problem if the level-1 identifier (i.e., the variable defining the units at level 1) does not have consecutive numbers. The only requirement is that all data for each group are contiguous in the data and that, within each group, the unit identifier increases as you go "down the data". The MLBOOK1.DAT data set is correctly ordered in this way.

Above, we already defined the "constant variable" *cons*. Therefore we have all we need to fit the simplest multilevel model: the empty model. (Wow!) To do this, go to the Equations window. What is in red is undefined, the black symbols are defined. On the bottom of this window you see a tool bar. You may click on the Name button to replace the symbols y etc. by their names. Click again, and the names are replaced by the symbols again. Now click on the red x_0 . In the window that pops up, select *cons* as the desired variable and in this window check the boxes Fixed parameter, j (school), as well as i (pupil). This means, respectively, that the effect of this variable is put in the fixed part, also in the random part at the school level, and also at the random part at the pupil level. Clicking Done will show the result in the equations window. Now you can click the + button at the tool bar. The + and − buttons determine the amount of detail displayed. Play around with them to have the maximum amount of detail. Now click the Estimates button and the symbols that refer to the parameters of the statistical model will light up in blue. Clicking this button once more will replace the symbols by blue numbers. These are their provisional values. They are blue, indicating that they have not yet been estimated from the data.

Now you are in the position to estimate the parameters. Click on the Start button near the upper left corner of the screen and MLwiN starts calculating. You will see the movement! After a little bit of time the iterative estimation algorithm has converged and you can see the estimates in the window. If you don't see them, click the Estimates button until you do. The estimates are given, followed by their standard errors in parentheses. You will see something like this.

$$\text{langpost}_{ij} \sim \text{N}(XB,\ \Omega)$$
$$\text{langpost}_{ij} = \beta_{0ij}\text{cons}$$
$$\beta_{0ij} = 40.364(0.426) + u_{0j} + e_{0ij}$$
$$\begin{bmatrix} u_{0j} \end{bmatrix} \sim \text{N}(0,\ \Omega_u) \ :\ \Omega_u = \begin{bmatrix} 19.421(2.923) \end{bmatrix}$$
$$\begin{bmatrix} e_{0ij} \end{bmatrix} \sim \text{N}(0,\ \Omega_e) \ :\ \Omega_e = \begin{bmatrix} 64.569(1.966) \end{bmatrix}$$

*-2*loglikelihood(IGLS Deviance)* = 16253.220(2287 of 2287 cases in use)

You can get this in e.g. a Word file in the following way. In MLwiN, press the <Ctrl>-C key in this window; alternatively, click on the Edit window and select Copy; then go to the Word file and press <Ctrl>-V, or alternatively select Edit and then Paste Special. You can resize the graphical image that you have imported in this way in your Word file by clicking on the picture, moving your mouse to one of the 8 points on the border, and moving the point inward or outward. Another way to retain this result is to print it by using the Print Window Image item in the File menu, or to use a log file (see below at the end of the next section).

Try to get the meaning of all symbols and numbers! Note that *cons* is the variable equal to 1; this variable is used to represent the intercept. With Ω is meant the covariance matrix of the random part, split here in the random part for level two (represented by Ω_u) and the random part for level one (represented by Ω_e). You see that the fixed effect (also indicated by MLwiN as β_0) is 40.364 (s.e. 0.426), the level-1 variance is 64.569 (s.e. 1.967) and the level-2 variance is 19.419 (s.e. 2.921). The deviance (= minus twice log-likelihood, see Section 6.2 of Snijders and Bosker, 1999) is 16253.220. You can go to Table 4.1 in Snijders and Bosker (1999) to see the same results reported.

You can also see the estimates in another window. Select the Model – Estimate tables window and you shall see one, or some, of the estimates being displayed. With the little + and − buttons in the upper left corner of this window you can add estimate displays. One of the displays has a bold outline, this is the *active* display. The parameter in this display can be selected in the drop down list (choosing between FIXED PART, Level 1: pupilnr, and Level 2: schoolnr, the latter referring to the random part), while the amount of detail can be determined by the check boxes. You shall find out how this works by playing around with it. The fact that the number in one of the fields has a line through it is no error: this is the previous value of the estimate in the iterative procedure, and it has a line to stress that it is not a correct outcome of the estimation process! It can be included in this table to give you the possibility of checking what happened in the iterative process. You can specify the model in the Equations window or in the Estimate Tables window, whichever you find more convenient.

Instead of pasting either of these windows into your Word (or other) file, you can also get the parameters in the log file. Get the Output window so that it is visible again. Now click again on Start. To get a listing of the parameter estimates, go to the Command Interface and first issue the command fixe, then

the command rand, and then like. An alternative is to use the Macro Editor to make a little macro with these three commands on consecutive lines, and Execute this macro. This will list the values of the parameters of the fixed part, of the random part, and the deviance, referred to as "–2⋆log(lh)" (i.e., minus twice the log-likelihood) in the Output window. Have a look at this window: what is the estimate of the fixed constant effect, what are the intercept variance and the residual variance, what are the standard errors of all these estimates? It may be convenient to arrange the Output window and the Equations window next to each other to have the best opportunity to compare them and understand what is what. Note that the Output window is saved only if you have earlier given the logon command.

3 Next steps of fitting a multilevel model

Continuing our session we wish to examine, e.g., the effect of *iq_verb* = verbal intelligence on language achievement. First we like to have some initial data description. Basic statistics (averages etc.) are computed by using the Basic Statistics – Averages and Correlation menu item and looking up the result in the Output window. This yields the results

```
                N     Missing     Mean          s.d.
iq_verb      2287           0     6.2456e-005   2.0689
```

Note that the mean was earlier subtracted from this variable. The notation `6.2456e-005` means 0.000062456, i.e., 6.2456 multiplied by 10 to the power −5, which practically equal to 0.

For the dependent variable *langpost* we get

```
                N     Missing     Mean          s.d.
langpost     2287           0     40.935        9.0037
```

Now we resume modeling. We had gone only as far as fitting the empty model. Now let us include *iq_verb* as an explanatory variable. Go to the Equations window and, if necessary, press the + button repeatedly to see the maximum amount of detail of the model. Now click the Add Term button. A window pops up asking for the Order of the new variable; for main effects the order is 0, so select order 0 and as variable select *iq_verb*. Confirm this selection (by pressing the Done button) and you will see that in the Equations window the variable IQ_verb$_{ij}$ is added (or x_{1ij} if the Name option is not selected). Since MLwiN detected that this is a level-one variable (it depends on the pupil), it has the double index ij representing that this is a variable depending on the pupil i and the school j. You can try this feature out by clicking on this variable again and changing it into *schoolses*, the mean *ses* (socio-economic status) in the school. Since this is a school variable, after confirming this choice it is represented by schoolSES$_j$ or x_{1j} as it depends only on the school, j. But now let us change this variable again and work with a fixed part consisting only of the constant cons = x_0 (do you realize now why this 'variable' has no subscript?) and *iq_verb* as x_1 . Now click on Estimates (if necessary, repeatedly) to see the numeric values of the estimates. They are blue because they were not obtained as a result of fitting this model; indeed, these are still the earlier obtained values. Click on More (near the upper left corner of the screen) to let the program

start estimating. (The difference between the Start and More buttons is that the More button uses the current parameter estimates as the initial values for the iterative algorithm, whereas Start starts from zero estimates.) You will see the numbers changing as the algorithm goes through its iteration steps until convergence. The result is given below. Compare this to Table 4.2 in Snijders and Bosker (1999). You see that we rounded the figures to two decimals, but the results further are identical.

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

$$\text{langpost}_{ij} = \beta_{0ij}\text{cons} + 2.488(0.070)\text{iq\_verb}_{ij}$$

$$\beta_{0ij} = 40.609(0.307) + u_{0j} + e_{0ij}$$

$$\begin{bmatrix} u_{0j} \end{bmatrix} \sim \text{N}(0,\ \Omega_u) \ :\ \Omega_u = \begin{bmatrix} 9.497(1.516) \end{bmatrix}$$

$$\begin{bmatrix} e_{0ij} \end{bmatrix} \sim \text{N}(0,\ \Omega_e) \ :\ \Omega_e = \begin{bmatrix} 42.227(1.286) \end{bmatrix}$$

*-2*loglikelihood(IGLS Deviance)* = 15251.770(2287 of 2287 cases in use)

The regression coefficient of *iq_verb* is estimated as 2.488, with standard error 0.070. The associated t-statistic is $2.488/0.070 = 35.54$, quite significant. The significance of the fixed effect of this variable can also be tested by a deviance test: the deviance went down from 16253.220 to 15251.770, a tremendous decrease of 1001.45 which here is a chi-squared value with one degree of freedom (because only one parameter was added to the statistical model). Furthermore, both the intercept variance and the residual variance have decreased considerably.

Besides pasting the Equations window into your text processor, there is another way to keep track of the results of successive model fits. If you go to the Command Interface and give the commands fixe, rand, and like (or do this through a macro in the Macro Editor, as mentioned above) while your log file is still on, you will have the parameter estimates reproduced in the logfile (if logging is on!!) in the following way (there may be differences in the last decimals):

```
fixe

PARAMETER            ESTIMATE     S. ERROR(U)   PREV. ESTIMATE
cons                    40.61       0.3069             40.61
iq_verb                 2.488      0.07005             2.488

rand

LEV.  PARAMETER       (NCONV)    ESTIMATE    S. ERROR(U)  PREV. ESTIM     CORR.
-------------------------------------------------------------------------------
 2    cons     /cons     ( 1)       9.496          1.515        9.511          1
-------------------------------------------------------------------------------
 1    cons     /cons     ( 2)       42.23          1.286        42.22

like

-2*log(lh) is      15251.8
```

Compare these figures with the figures in the Equations window and also with Table 4.2 of the book to see how everything corresponds. Also go to the

Estimate Tables window to see the same estimates presented in a different way. Pressing the Help button in that window gives you excellent help on how to operate it.

3.1 Estimation methods

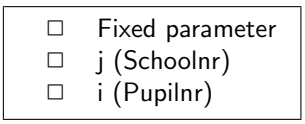
There are several statistical methods to carry out estimation in multilevel models. If you push the button Estimation Control you see a window in which you can control these methods. The two standard methods are called IGLS (iterated generalised least squares) and RIGLS (residual, or restricted, igls) in the MLwiN jargon. The IGLS method yields maximum likelihood estimates. RIGLS is in the literature also called residual or restricted maximum likelihood, REML. Checking the RIGLS field changes the estimation method to REML. For small sample sizes, REML is somewhat preferable as an estimation method. Testing variance components by deviance tests, however, proceeds more easily with unrestricted maximum likelihood, which is the same as the IGLS method. The reason is that deviance differences from RIGLS fits can be used as test statistics only for models that have the same fixed parts. Deviance differences from IGLS model fits (provided that one model is a strict submodel of the other) can always be used as test statistics. With the RIGLS method, the estimates produced in the logfile by the commands fixe and rand are as follows:

```
fixe
PARAMETER            ESTIMATE     S. ERROR(U)   PREV. ESTIMATE
cons                    40.61       0.3081             40.61
iq_verb                 2.488      0.07008             2.488

Rand

LEV.  PARAMETER       (NCONV)    ESTIMATE    S. ERROR(U)  PREV. ESTIM     CORR.
-------------------------------------------------------------------------------
 2    cons     /cons     ( 0)       9.594          1.516        9.498         1
-------------------------------------------------------------------------------
 1    cons     /cons     ( 3)       42.25          1.286        42.23
```

There is not much difference in this case. In more complicated models there often are bigger differences.

3.2 Which boxes to check when adding new variables to the model

A crucial part in many MLwiN sessions is adding new variables to the model. In the present session, we added *cons* and *iq_verb*. The default is that variables get a fixed effect only. To include in the model also a random effect, click on this variables in the Equations window. Then a pop-up window appears for this variable with three boxes that could be checked:



The first indicates the fixed effect, the next two indicate the random effects at level 2 (*Schoolnr*) and 1 (*Pupilnr*), respectively. If you would have a three-level model, there would be three boxes for the random effects.

You select contributions to the model by checking these boxes. The default is the following:

- for the constant (*cons*) you select all three, because these represent the general constant term in the fixed part, the random intercept, and the level-1 residual, respectively;
- for all variables you select the fixed effect;
- for level-two variables you select only the fixed effect and no random effects;
- for level-one variables you select the fixed effect and, if you want to include a random slope in the model (which depends on theory & data), you also select the random effect at level 2;
- however, for interactions including level-one variables (represented by product variables), you usually select only the fixed effect and not the random effect.

These rules are no more than a first guideline, and in more complicated models there appear many exceptions to them. But they are adequate for a start in multilevel modeling.

When you wish to delete a variable from the model, do not do this by unchecking all three boxes, but by clicking on the delete Term button. (Unchecking all boxes make you lose the variable, and you can put it back again only through the Command Interface.)

3.3 Random slope models and interaction variables

Next, we go on with modeling. We wish to go on to defining random effects and interactions. We wish now to reproduce Table 5.1 of Snijders and Bosker (1999). The variable *iq_verb* is given a random effect; more precisely, this effect is random at level 2 (i.e., it differs from school to school). This is done by going to the Equations window, clicking on the x_1 variable (representing *iq_verb*) and there checking the j (school) box. Now you can see that the coefficient is represented by β_{1j} to express its dependence on the school j. Pressing the More button will carry out the estimations. The deviance decreases from 15251.8 to 15230.8, a decrease of 21.0 for 2 degrees of freedom (there are two extra parameters: the slope variance and the slope- intercept covariance), highly significant again.

3.3.1 More multilevel data manipulation

Now we also have to include the school (rather, classroom) mean of IQ. However, this variable is not yet in the data set; only individual IQ is. We calculate the school mean in the Data Manipulation – Multilevel Data Manipulations window by choosing Operation Average, Input column iq_verb on blocks defined by schoolnr, and selecting Free column *C33* . When this command has been added to the action list and executed, we can name the new variable *C33* in the Names window as *sch_iqv*.

After some experience with MLwiN, users may find it quicker to use the Command Interface or a macro with the two commands

```
mlav C1 "iq_verb" C33
name C33 "sch_iqv"
```

The command MLAVerage calculates the group averages, where the groups are defined here by variable *C1 = schoolnr*.

Adding this variable to the model with a fixed effect only and estimating the parameters yields the following estimates: (if your results are slightly different, you probably forgot to set the RIGLS estimation method back to the IGLS method; if they are very different, something may have gone wrong with centering the variables: check if *sch_iqv* has average 0, as it should!).

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

$$\text{langpost}_{ij} = \beta_{0ij}\text{cons} + \beta_{1j}\text{iq\_verb}_{ij} + 1.405(0.321)\text{sch\_iqv}_j$$

$$\beta_{0ij} = 40.750(0.286) + u_{0j} + e_{0ij}$$

$$\beta_{1j} = 2.459(0.083) + u_{1j}$$

$$\begin{bmatrix} u_{0j} \\ u_{1j} \end{bmatrix} \sim \text{N}(0,\ \Omega_u) \ :\ \Omega_u = \begin{bmatrix} 7.921(1.317) & \\ -0.822(0.268) & 0.200(0.098) \end{bmatrix}$$

$$\begin{bmatrix} e_{0ij} \end{bmatrix} \sim \text{N}(0,\ \Omega_e) \ :\ \Omega_e = \begin{bmatrix} 41.350(1.287) \end{bmatrix}$$

*-2*loglikelihood(IGLS Deviance)* = 15213.530(2287 of 2287 cases in use)

Note that we obtain a 2×2 covariance matrix at level 2: the random intercept variance (7.921), the slope-intercept covariance (−0.822), and the random slope variance (0.200). Compare these results to Table 5.1. The estimates can also be inspected in the Estimate tables window. The last decimals can differ due to lack of computational precision.

From the preceding we learn the following: whenever you have estimated a model and wish to retain the results, take care that the log file is on and issue the commands: fixe, rand, like; or paste the results from some relevant window in a file, using (e.g.) Word. Output will not be saved in some automatic way!

Now let's go on and estimate a cross-level interaction effect to try to "explain" the random slope. (Recall that for the definition of product variables to represent interaction, it is convenient for the interpretation that the "0" value is within the range – or otherwise meaningful – of each of the factors in the product. In this case, we use the level-2 variable *groupsize*, which is centered just like *iq_verb*, i.e., their "0" value is their mean.) First add the main effect of groupsize to the model.

There are two ways to specify the interaction effect. One way is to define the interaction variable by the calculation

```
c34 = "iq_verb"*"groupsiz"
```

give it a name, e.g., *z2*iq*, and add it to the model (with Order 0 just like above).

The other way is to open the Add Term dialog box, but now choose Order 1. This implies you wish to specify a first-order interaction effect as a product of two variables. After doing this, two fields will be opened for variable

specifications. Choose the variables *groupsize* and *iq_verb*. Since these variables are continous (rather than categorical), you will not be requested to specify reference categories.

When the main and interaction effects have been added to the model, estimate the parameters again. If you wish to, you can decrease the font size in the display (use the Fonts button in the Equations window). The result is as follows.

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

$$\text{langpost}_{ij} = \beta_{0ij}\text{cons} + \beta_{1j}\text{iq\_verb}_{ij} + 1.246(0.326)\text{sch\_iqv}_j + -0.022(0.011)\text{z2*iq}_{ij} + 0.057(0.037)\text{groupsiz}_{ij}$$

$$\beta_{0ij} = 40.893(0.292) + u_{0j} + e_{0ij}$$

$$\beta_{1j} = 2.443(0.082) + u_{1j}$$

$$\begin{bmatrix} u_{0j} \\ u_{1j} \end{bmatrix} \sim \text{N}(0,\ \Omega_u) \ :\ \Omega_u = \begin{bmatrix} 7.668(1.285) & \\ -0.769(0.260) & 0.178(0.095) \end{bmatrix}$$

$$\begin{bmatrix} e_{0ij} \end{bmatrix} \sim \text{N}(0,\ \Omega_e) \ :\ \Omega_e = \begin{bmatrix} 41.362(1.287) \end{bmatrix}$$

$$-2*\textit{loglikelihood(IGLS Deviance)} = 15208.390(2287 \text{ of } 2287 \text{ cases in use})$$

(You may be surprised that the variable *groupsize* has a double index, with i as well as j. This is because the nesting here is on schools, and the data set contains some schools with more than one group.) These results are just like in Table 5.2. You can select a smaller font size in the Equations window if you wish.

Another way to change the model is provided in the Estimate Tables window. You can change the variables in the "active window" (which may represent either the fixed part, or the random part at level 1, or the random part at level 2) (defined above) by clicking the ± box, and then checking the variables that you want to be included in this part of the model. This works faster if you want to change more than one variable at a time.

By now we have changed so much that it is useful to save the results again in the worksheet like it was mentioned above. In general, it is wise to do this repeatedly to save you from annoyance when some error occurs due to a bug in MLwiN or instability of Windows.

The results of the random coefficient multilevel model can be compared to results of "ordinary" regression analysis (where everything is disaggregated to the lowest level) by issueing the command olse ("ordinary least squares estimates") in the Command Interface. It yields the results

```
  PARAMETER            ESTIMATE     S. ERROR
cons                      40.95       0.1472
iq_verb                   2.422      0.07766
sch_iqv                   1.344       0.1991
z2*iq                   -0.01805     0.009826
groupsiz                 0.05027      0.02039

SIGMA SQUARED =       49.41
```

Especially standard errors obtained by ordinary regression are not trustworthy. They often are too low. This especially holds for the school variables.

4 Posterior means, or level-two residuals

The posterior means of the level-2 units can be requested in the Model – Residuals window. To keep things simple, first specify a model with only the fixed effect of *iq_verb* (click on the variables in the Equations window, choose Delete for all other variables, and for *iq_verb* choose only the fixed effect). Estimate this model again. Now go to the Residuals window, where you will come in the Settings tab. To keep things simple and not get ununderstood output, uncheck all the Normal scores etc. boxes (unless you know what they mean). Select level: 2:schoolnr to get the posterior means for the schools, which are just the school-level residuals. Press the Set columns button. The residuals, their standard errors, and the standardised residuals will be put into columns *C300-C302* unless you indicate a different starting column. When you press Calc the calculations will be carried out. In the Data window you can view the residuals and their standard errors. To get them in your logfile, go to the Command interface and issue the command

```
print C300-C302
```

I only gave the command

```
print C300
```

which yields output starting with

```
        c300
 N =     131
   1  -0.37548
   2  -6.0197
   3  -3.6468
   4  -2.9075
   5  -5.7224
   6  0.80680
   7  -6.3489
   8  -1.3254
   9  3.4082
  10  -1.0165
  11  -5.0541
```

This means that this column has 131 elements (there are, indeed, 131 schools) and that, e.g., the posterior mean for school 5 (the estimated value for residual U_5) is -5.7224. To get the posterior means (or level-2 residuals) for the random slope model with a random slope for *iq_verb*, add the random slope for this variable to the model (you should know by now how to do this) and estimate parameters again. Go again to the Residuals window, and specify the Settings tab as before. Now you will see that MLwiN is going to put the residuals in columns *C300* and *C301* , since there are indeed two residuals: the random intercept and the random slope for *iq_verb*. Pressing the Calc button and issueing the command print C300-C301 now gives output starting with

```
        c300          c301
N =      131           131
  1   -0.29756      -0.020446
  2   -5.0311       0.63247
  3   -3.8606       0.54635
  4   -2.3994       0.30820
  5   -5.8617       0.75835
  6   0.64810       -0.063412
  7   -6.4978       0.84366
  8   -1.4007       0.17338
  9   3.3087        -0.38413
```

These are the estimated values of U_{0j} and U_{1j}, i.e., the posterior means of the contributions of the schools to intercepts and slopes. The intercepts and slopes are

Predicted values are calculated in a similar way in the Prediction window. Using the Help facility, you may find out by yourself how this works.

MLwiN provides two different types of standard errors of the residuals / posterior means: comparative and diagnostic standard errors. The diagnostic standard errors are bigger. These are the standard errors of the estimated residuals about their population mean of 0. They are useful for diagnostic checking, e.g., when checking normality of the distribution of the random effects (e.g., Snijders & Bosker, 1999, p. 132–133). The comparative standard errors are the standard deviation of the difference between the estimated residual and the (unknown true) random effect. They are useful for comparing groups (level-two units) with respect to their random effect values (e.g., Snijders & Bosker, 1999, p. 60–63). In the Residuals window, it is shown that next to residuals themselves, always the comparative standard errors are calculated as well as the standardised residuals based on the diagnostic standard errors, i.e., using the definition

$$\text{standardised residual} \;=\; \frac{\text{residual}}{\text{diagnostic standard error}} \;.$$

The "caterpillar plots" of residuals with error bars, as presented in Figure 4.4 of Snijders and Bosker (1999), can be made as follows. In the Residuals window, also select the ranks of residuals, and to have (as in Figure 4.4) error bars extending up and down for a length of 1.39 times the comparative standard error, type the number 1.39 in the box before SD (comparative) of residual to. After having pressed the Calculate button, open the Plots tab, and select the option single .. residual +/- 1.36 sd x rank. After clicking Apply, for each random effect you get a plot which you can increase in size if you wish, and copy and paste to use in other programs.

Plots of estimated regression lines can be obtained in a different way. Suppose you wish regression lines as a function of *iq_verb*. Then you must specify a model with a random intercept, with or without a random slope for *iq_verb*. It can contain other level-two variables, but if you include any other level-1 variables, it is best to choose variables for which 0 is a meaningful value, so that

it will be meaningful to plot the regression lines in which these other values are set equal to 0.

First estimate the posterior means as above. Then open the Model - Predictions window. In the window that you now see, select *cons*, iq_verb, and the other level-2 variables (if any). For *cons*, deselect the level-1 residual e_{0ij}. This means that you will have selected a number of fixed effects and one or two level-two random effects, and *not* the level-1 residual. The selected components are added to form the prediction. Select a column in which to put the output from the prediction, e.g., *c51*. Now open the Graphs – Customised Graph(s) window. For y choose the predicted values – in this example *c51* –, for x the variable *iq_verb*, choose plot type – line and group – *schoolnr*. The latter specification means that for each group as defined by *schoolnr*, the consecutive plotted values (in the order defined by the horizontal variable x) are linked by straight lines. Now click on Apply and the plot will be exhibited.

5 Macro editor

The macro editor can be accessed through the File menu. An easy introduction is to use the macro CH458.OBE which is also contained in the mlbook.zip file (this macro has also been called MLBOOK1.OBE in some versions). Unpack the file CH458.OBE from the file mlbook.zip and put it in the "current directory" of MLwiN(chosen through Options – directories) where you must also put the file MLBOOK1.DAT.

Then in the File menu, choose Open macro, go to this directory, select Files of type: all files (*.*), and then open CH458.OBE.

This opens the file in the macro editor. The macro editor has the nice feature that MLwiN commands (more precisely, their significant four first letters) are indicated in blue, notes in green, data and variables in black, and errors (guess the color). To see the output of the macro, open the output window of the command interface open (Data Manipulation – Command interface).

Take care that currently, you are not making a log file. To execute the macro, press the Execute button in the lower left corner of the macro editor. Look in the output window to see the results. In the current MLwiN directory, also the file mlbook1.log will have been created.

To understand the macro, you must understand the meaning of the commands. This macro uses, e.g., the following commands:

echo	echo commands to the log file and the output window
logo 0	terminate earlier log file, it is was there
logo	start new log file
clea	delete the earlier model specification
note	disregard this line
obey	execute a macro
code	define new integer variable
name	give a name to a variable
dumm	construct dummy variable

take	take the first value from each group to define a new group-level variable
tabu	tabulate
aver	calculate the average
calc	calculate a new variable
pref 0	needed for batch processing
post 0	also needed for batch processing
iden	define level identifying variable
expl 1	define an explanatory ('independent') variable
setv 1	give a variable a random effect at some level
resp	define the dependent variable
batc	carry out subsequent estimations in batch (without clicking)
maxi	maximum number of iterations per batch estimation
star	start estimation
fixe	report fixed parameter estimates
rand	report parameter estimates in random part of the model
like	report deviance
next	continue estimation
clrv	delete a random effect at some level
olse	report OLS estimates
sete	add an element to the random effects covariance matrix at some level

A more systematic list is in Section 9. To understand the syntax of these commands, you can look them up in the MLwiN Command Manual. Alternatively, look in the Help facility for Command echo, Command logo, etc.

6 Heteroscedasticity

Heteroscedasticity is the same as non-constant variances, and sometimes referred to as complex variation. One of the nice features of MLwiN is that it gives the very straightforward possibility to let variances depend linearly or quadratically on explanatory variables. This can be used also if single-level heteroscedastic models are to be fitted.

When the following example is done after fitting the macro of the preceding section, start by issuing the command clear to get rid of the earlier model specification.

As an example, estimate a baseline (homoscedastic) model with fixed effects for *iq_verb*, *sch_iqv*, *SES*, and *sex*; and with a random slope for *iq_verb*. These are the results reported as Model 1 in Table 8.1 in Snijders & Bosker (1999).

$$\begin{aligned}
&\text{langPOST}_{ij} \sim \text{N}(XB,\ \Omega)\\
&\text{langPOST}_{ij} = \beta_{0ij}\text{cons} + \beta_{1j}\text{IQ_verb}_{ij} + 1.023(0.322)\text{schIQV}_j +\\
&\qquad\qquad 0.152(0.014)\text{ses}_{ij} + 2.644(0.264)\text{sex}_{ij}\\
&\beta_{0ij} = 39.530(0.313) + u_{0j} + e_{0ij}\\
&\beta_{1j} = 2.268(0.081) + u_{1j}
\end{aligned}$$

$$\begin{bmatrix} u_{0j} \\ u_{1j} \end{bmatrix} \sim \text{N}(0,\ \Omega_u) \ :\ \Omega_u = \begin{bmatrix} 8.275(1.331) & \\ -0.763(0.254) & 0.169(0.087) \end{bmatrix}$$

$$\begin{bmatrix} e_{0ij} \end{bmatrix} \sim \text{N}(0,\ \Omega_e) \ :\ \Omega_e = \begin{bmatrix} 37.555(1.170) \end{bmatrix}$$

*-2*loglikelihood(IGLS Deviance)* = 15005.490(2287 of 2287 cases in use)

Now go the Equations window and click on the variable *iq_verb*. In the drop-down menu, select i(Pupilnr), which means that this variable gets a random effect at level 1. As explained in Section 8.1.2 of Snijders & Bosker (1999), this means that the level-1 residual variance is a quadratic function of *iq_verb*. Upon clicking done, the Equations window displays a two-by-two covariance matrix also for level 1. Estimating the model gives the following result.

$$\begin{aligned}
&\text{langPOST}_{ij} \sim \text{N}(XB,\ \Omega)\\
&\text{langPOST}_{ij} = \beta_{0ij}\text{cons} + \beta_{1ij}\text{IQ_verb}_{ij} + 1.012(0.321)\text{schIQV}_j +\\
&\qquad\qquad 0.146(0.014)\text{ses}_{ij} + 2.515(0.258)\text{sex}_{ij}\\
&\beta_{0ij} = 39.602(0.310) + u_{0j} + e_{0ij}\\
&\beta_{1ij} = 2.215(0.077) + u_{1j} + e_{1ij}
\end{aligned}$$

$$\begin{bmatrix} u_{0j} \\ u_{1j} \end{bmatrix} \sim \text{N}(0,\ \Omega_u) \ :\ \Omega_u = \begin{bmatrix} 8.083(1.310) & \\ -0.529(0.241) & 0.148(0.080) \end{bmatrix}$$

$$\begin{bmatrix} e_{0ij} \\ e_{1ij} \end{bmatrix} \sim \text{N}(0,\ \Omega_e) \ :\ \Omega_e = \begin{bmatrix} 38.167(1.313) & \\ -1.845(0.292) & -0.121(0.156) \end{bmatrix}$$

*-2*loglikelihood(IGLS Deviance)* = 14959.500(2287 of 2287 cases in use)

The interpretation is that the level-1 variance is

$$\text{var}(R_{ij}) \ = \ 38.167 \ - \ 3.690\, \textit{iq\_verb} \ - \ 0.121\, \textit{iq\_verb}^2$$

(note that, as explained in Section 8.1.2, the covariance term must be multiplied by 2). It does not matter that the 'variance' parameter is negative, as long as the estimated variances for the dependent variable Y_{ij} are always positive.

A linear variance function is obtained by deleting the level-1 variance parameter. This is done by clicking on it (i.e., on the number -0.121) and replying

by 'yes' to the question whether to remove this term from the level-one covariance matrix. This fixes this parameter to 0. Estimating the model yields the following.

$$\mathbf{langPOST}_{ij} \sim \mathrm{N}(XB,\ \Omega)$$
$$\mathbf{langPOST}_{ij} = \beta_{0ij}\mathbf{cons} + \beta_{1ij}\mathbf{IQ\_verb}_{ij} + 1.017(0.321)\mathbf{schIQV}_{j} + 0.146(0.014)\mathbf{ses}_{ij} + 2.510(0.258)\mathbf{sex}_{ij}$$
$$\beta_{0ij} = 39.608(0.310) + u_{0j} + e_{0ij}$$
$$\beta_{1ij} = 2.223(0.077) + u_{1j}$$

$$\begin{bmatrix} u_{0j} \\ u_{1j} \end{bmatrix} \sim \mathrm{N}(0,\ \Omega_u) \ :\ \Omega_u = \begin{bmatrix} 8.061(1.307) & \\ -0.509(0.240) & 0.133(0.077) \end{bmatrix}$$

$$\begin{bmatrix} e_{0ij} \\ e_{1ij} \end{bmatrix} \sim \mathrm{N}(0,\ \Omega_e) \ :\ \Omega_e = \begin{bmatrix} 37.826(1.195) & \\ -2.013(0.257) & 0 \end{bmatrix}$$

*-2*loglikelihood(IGLS Deviance)* = 14960.040(2287 of 2287 cases in use)

The increase in deviance, 0.54, is not significant (one parameter is involved, hence *d.f.* = 1). The new variance function is

$$\mathrm{var}(R_{ij}) = 37.826 - 4.026\, iq\_verb\ .$$

This result is also given in Table 8.2 in Snijders & Bosker (1999). To reproduce Model 4 of this table, the new variables IQ^2_- and IQ^2_+ can be calculated by the following macro. The function abso is the absolute value. Try to understand the calculation of the new variables *C38* and *C39* by writing out the implied equations separately for $iq\_verb > 0$ and $iq\_verb < 0$. The last two plot commands are meant only to visually confirm the calculation of these two 'half squares'.

```
calc c37 = 'iq_verb'^2
name c37 'iqv^2'
calc c38 = 'iq_verb'*abso('iq_verb')
calc c38 = (c38 + 'iqv^2')/2
name c38 "iqv^2+"
calc c39 = c37 - c38
name c39 "iqv^2-"
plot c38 c3
plot c39 c3
```

Estimating the model with these two variables added gives the following result. For the interpretation, see page 113 of Snijders & Bosker (1999).

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

$$\begin{aligned}\text{langPOST}_{ij} = {} & \beta_{0ij}\text{cons} + \beta_{1ij}\text{IQ\_verb}_{ij} + 1.212(0.308)\text{schIQV}_j + \\ & 0.143(0.014)\text{ses}_{ij} + 2.354(0.254)\text{sex}_{ij} + -0.306(0.039)\text{iqv\^{}2+}_{ij} + \\ & 0.246(0.046)\text{iqv\^{}2-}_{ij}\end{aligned}$$

$$\beta_{0ij} = 39.728(0.310) + u_{0j} + e_{0ij}$$

$$\beta_{1ij} = 3.236(0.157) + u_{1j}$$

$$\begin{bmatrix} u_{0j} \\ u_{1j} \end{bmatrix} \sim N(0, \Omega_u) : \Omega_u = \begin{bmatrix} 7.189(1.173) & \\ 0.000(0.000) & 0.000(0.000) \end{bmatrix}$$

$$\begin{bmatrix} e_{0ij} \\ e_{1ij} \end{bmatrix} \sim N(0, \Omega_e) : \Omega_e = \begin{bmatrix} 37.875(1.181) & \\ -2.370(0.225) & 0 \end{bmatrix}$$

*-2*loglikelihood(IGLS Deviance)* = 14908.060(2287 of 2287 cases in use)

7 Assumption checking

In Chapter 9 of Snijders & Bosker (1999), various methods for assumption checking are discussed. The macros made available at the website of the book give an easy possibility to replicate some of the examples. In this section, examples are given of some general-purpose macros that can be used for assumption checking. These are the macros RES1.OBE, TABLE.OBE, and DINFL.OBE that also are obtainable from the website of the book. It will be convenient to have these macros in the working directory of your MLwiN session.

7.1 Level-1 OLS residuals

The specification of the level-1 model can be checked by using level-1 residuals estimated by OLS (ordinary least-squares regression) for each group (i.e., each level-2 unit) separately. This can be done using the RES1.OBE macro. As an example, it will be shown how this macro can be used together with TABLE.OBE to reproduce the left panel in Figure 9.1, which is the average level-1 OLS residual as a function of *iq_verb*. (The word 'average' is used because *iq_verb*, although approached as a continuous variable, has only a limited number of categories, and it is best to average the residuals within each of these categories).

As a first step, specify the model with as usual the dependent variable *langpost*, with the level-1 explanatory variables *iq_verb, sex*, and *ses*, and so that it has *no level-2 explanatory variables* – otherwise the level-1 OLS residuals are not meaningful. Estimate this model (the deviance should come out as 15032.620). It will be convenient to save the worksheet at this point, in case anything goes wrong later on.

In the File – Open Macro menu, open the macro RES1.OBE. This macro

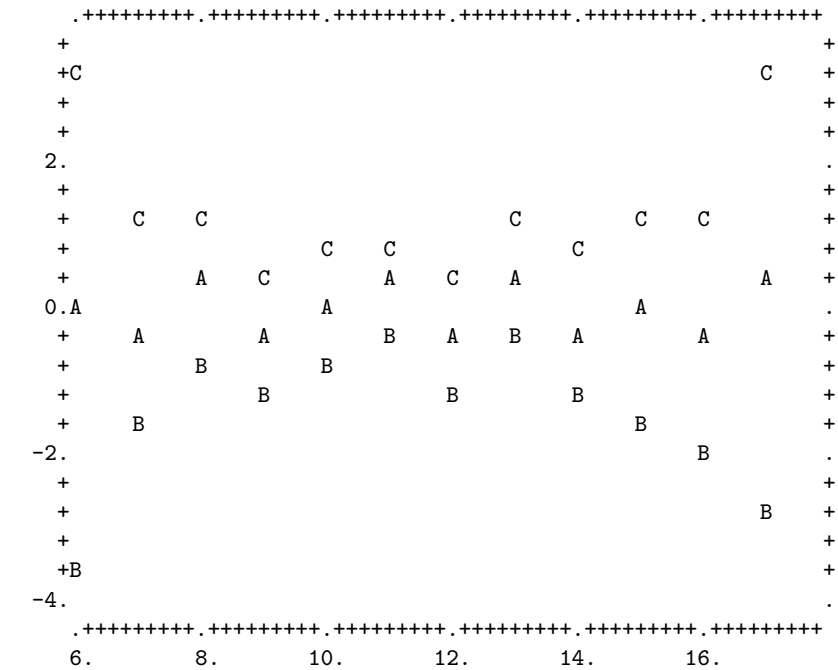
calculates the level-1 OLS regressions for all groups with a group size at least equal to some constant value, of which the default is 10 (this can be changed by setting the box *B11* to a different value, see Command set in the Help facility). Take care that the Output window is open, so you can see what the macro tells to the user, which are the names and column numbers of the variables calculated. It will also be helpful if logging is on, so that the results of the output window are saved in the log file. The (non-standardized) level-1 OLS residuals are in column *C226*. A reduced data set is constructed – reduced because the groups of size less than 10 are omitted. The explanatory variables in this reduced data set are in group *G4*. In the Data manipulation – Groups menu, you can see that group *C4* has columns *C301-C303*, corresponding to group textitG1 with variables *iq_verb, sex, ses* in the total data set.

To average the level-1 residuals for each group of *iq_verb*, the TABLE.OBE macro is used. Open this macro in the macro editor. The start of the macro explains that it calculates averages over categories. It requires the column numbers of the variables, and the minimum size of each category. These are supplied by issueing the commands

```
set b1 301
set b2 226
set b3 10
```

which indicate that we wish to average *c226* in categories defined by the variable *c301* for groups of size 10 and larger.

When this macro has been executed, the output window (provided this was open during execution) tells that rounded values of $X = C301$ are in column *C211*, average values of $Y = C226$ are in column *C213*, and within-category standard deviations are in column *C214*. The following simple plot of the averages plus and minus two standard deviations is given in the output window.



This is like the left-hand panel of Figure 9.1 in Snijders & Bosker (1999), although in a less pretty format.

7.2 Level-two influence diagnostics

Next it is shown how diagnostic statistics can be calculated that express the influence of the level-two units on the parameter estimates. This is discussed in Section 9.6.2 of Snijders & Bosker (1999) and in Snijders & Berkhof (2007). Here, the macro DINFL.OBE is used, which calculates deletion diagnostics for the influence of level-2 units; the term 'deletion' means that for calculating the influence of each level-2 unit, parameter estimates are used computed after deleting this particular unit from the data set. This gives a better estimate of the influence than the method proposed in Snijders & Bosker (1999), which is entirely analogous except that the deletion principle is not used.

Influence will be calculated for Model 4 of Table 8.2. For this model, the two 'semi-squares' of *iq_verb* need to be calculated. This can be done by the commands

```
calc c30 = abso("iq_verb")*"iq_verb"
calc c31 = "iq_verb"*"iq_verb"
calc c32 = (c31+c30)/2
calc c33 = (c31-c30)/2
name c32 "iq+^2 c33 "iq-^2"
```

These new variables *c32* and *c33* are the same variables as IQ^2_+ and IQ^2_- defined on p. 113 of Snijders & Bosker (1999). Also the within-school average of *iq_verb* has to be calculated by the Multilevel Data Manipulations. Calculate these variables, give the *iq_verb* variable a random effect at level two, and fit the resulting model. The resulting model has parameter estimates represented as follows.

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

$$\begin{aligned}\text{langPOST}_{ij} = {} & \beta_{0ij}\text{cons} + \beta_{1j}\text{IQ\_verb}_{ij} + 2.571(0.261)\text{sex}_{ij} + 0.150(0.014)\text{ses}_{ij} + \\ & -0.327(0.050)\text{iq+}\hat{}2_{ij} + 0.248(0.042)\text{iq-}\hat{}2_{ij} + 1.031(0.314)\text{School\_IQ}_j\end{aligned}$$

$$\beta_{0ij} = 39.635(0.320) + u_{0j} + e_{0ij}$$

$$\beta_{1j} = 3.270(0.165) + u_{1j}$$

$$\begin{bmatrix} u_{0j} \\ u_{1j} \end{bmatrix} \sim \text{N}(0,\ \Omega_u) \ :\ \Omega_u = \begin{bmatrix} 7.934(1.277) & \\ -0.729(0.227) & 0.071(0.071) \end{bmatrix}$$

$$\begin{bmatrix} e_{0ij} \end{bmatrix} \sim \text{N}(0,\ \Omega_e) \ :\ \Omega_e = \begin{bmatrix} 37.171(1.155) \end{bmatrix}$$

*-2*loglikelihood(IGLS Deviance)* = 14959.700(2287 of 2287 cases in use)

When the macro DINFL.OBE is opened in the macro editor, it is seen that the macro starts with an explanation. Nothing extra needs to be prepared. It will be helpful, however, to ensure that the Output window is open and logging is on (as explained in Section 1.2). After clicking Execute, you will have to wait a few seconds until the calculations are ready. The output window gives several plots that can be used for diagnostic purposes (where Section 9.6.2 of Snijders

& Bosker (1999) and Snijders & Berkhof (2007) can be used as background material). The calculated variables are indicated in the last lines written in the output:

```
note  c204 is group identifier                                  (unit2_j)
note  c205 is group size                                        (n_j)
note  Groupwise influence statistics:
note  c201 = influence on random part parameters                (CR_j)
note  c202 = influence on fixed parameters                      (CF_j)
note  c200 = combined influence diagnostic                      (C_j)
note  c203 = standardized multivariate residual                 (S2_j)
note  (c203 approximately chi squared, d.f. in c205)
note  c206 = p-values for c203
note  c207 = observed normal deviates for c206
note  c208 = expected normal deviates for c206
note  Level-one influence statistics:
note  c209 = standardized level-1 group-deletion residuals (res_1)
```

These are available for further analysis. The most informative are the largest influence values, because these might point out ill-fitting level-two units. To get the largest values, we have to order these columns in descending order of *C200* $= C_j$. Ordering can be done using Data Manipulation – Sort but here it will be done in the Command Interface. MLwiN sorts in ascending order. To sort in descending order, we have to reverse the order of *C200* (and later put this back again). This is done by the next three commands:

```
calc c200 = -c200
sort c200 c201-c208 c200 c201-c208
calc c200 = -c200
```

In this way all variables *C200-C208* are sorted correspondingly. To inspect the main of these variables, issue the command

```
print c200 c204 c205 c206
```

The first 12 lines printed in the Output window are the following.

```
        C_j          unit2_j      n_j          p-val
 N =    131          131          131          131
   1  0.054952      176.00        23.000        0.13642
   2  0.046097      40.000        35.000        0.022137
   3  0.045150      142.00        24.000        0.18810
   4  0.040645      67.000        26.000        0.38442
   5  0.029258      147.00        22.000        0.70685
   6  0.027783      141.00        20.000        0.78782
   7  0.027476      170.00        26.000        0.0016163
   8  0.022898      256.00        10.000        0.072687
   9  0.022340      15.000        8.0000        0.0057100
  10  0.022144      108.00        9.0000        0.00010079
```

(The numbers are different from those in Table 9.1 of Snijders & Bosker, 1999, because macro DINFL.OBE calculates deletion residuals, which are treated in Snijders & Berkhof, 2007, but not in Snijders & Bosker, 1999.) This shows,

e.g., that the largest influence value is $C_j = 0.054952$ for school $j = 176$, which contains $n_j = 23$ pupils, and for which the p-value of the standardized multivariate residual is 0.13642. This influence value is not alarmingly large. Further the significance test of the fit of this level-two unit is not significant, as indicated by the p-value. Thus, there is no reason for big concern. On the other hand, school 108 has a very small p-value of 0.00010079 which does indicate a significant lack of fit, even with a Bonferroni correction, but the influence of this school on the parameter estimates is not overly large, and deleting this school from the data set would not lead to a dramatic change in the parameter estimates.

8 Multilevel logistic regression

There are various different way to carry out multilevel logistic regression in various software packages. Statistical theory still has not converged to a single best method for estimating parameters in multilevel logistic regression models. In this section we treat the simplest ways to estimate parameters in such models using MLwiN and illustrate these using examples in Chapter 14 of Snijders & Bosker (1999).

These examples use a different data set (a small part of the ISSP 1994 survey) than the previous sections. In order not to be confused with the results from earlier data sets, it is advised to start with a new MLwiN session for following this section. The data set is contained in the MLwiN macro IS9412.DAT, which also is comprised in the file MLBOOK.ZIP. The file IS9412.DAT is a macro that can be obeyed using the Command Interface (see Section 1.3.2) or the macro editor, which will result in importing the data set with the variable labels. When you have executed this macro, looking at Data Manipulation – Names will show that you have a worksheet with 10 variables and 2079 cases. This worksheet corresponds to the description in Example 14.1.

First we shall define the logistic regression model for this data set. The first step is that we must have available a total of *two*[6] variables all equal to 1 for all cases. Given that the data set already contains the variable C9 = *cons*, the quickest way to achieve this is by issueing the commands

```
calc c11 = c9
name c11 "denom"
```

The name *denom* is shorthand for *binomial denominator*. This variable indicates the number of "yes/no" cases for each level-one unit; in casewise (i.e., non-aggregated) data, this is equal to 1 for each case. The fact that *denom* = 1 for all level-one units thus implies that the dependent variable, which is the number of positive ("yes") cases for each unit, can have only the values 0 and 1. (For aggregated binomial data, this can be a variable with higher integer values.)

To define the model, take the following steps in the Model – Equations window:

- Choose (by clicking on the red y) the dependent variable *cohab* and specify 2 levels, the level 2 identifier being *reg* (region) and the level 1 identifier

[6]Those who used logistic regression in MLwiN versions earlier than 2.0 will note that there is a simplification, because the *bcons* variable is not needed any more, but is now made automatically by MLwiN.

being *respnr* (respondent).
Check in the Data Manipulation – Names window, that variable *cohab* indeed has values 0 and 1.

- Click on the **N** (for Normal distribution) letter and select Binomial in the little window that appears. In the additional little window, the logit link function is the right one. Click Done.
- Click on the red n_{ij} and select the variable *denom*.
- Click on the Nonlinear button in the bottom bar of the Equations window. Choose Use Defaults and then click on Done.
- Click on the red $\beta_0 x_0$ symbol, choose the *cons* variable, and select both the Fixed Parameter and j(reg) fields.
  This is like the use of *cons* for multilevel regression for normal distributed residuals, except that no level-one effect of *cons* needs to be, or can be, specified.

These steps should be sufficient to define the empty model for multilevel logistic regression. When you have clicked the Estimates and Names buttons in the Equations window, the window should look like this:

$$\text{cohab}_{ij} \sim \text{Binomial}(\text{denom}_{ij}, \pi_{ij})$$
$$\text{logit}(\pi_{ij}) = \beta_{0j}\text{cons}$$
$$\beta_{0j} = \beta_0 + u_{0j}$$
$$\begin{bmatrix} u_{0j} \end{bmatrix} \sim \text{N}(0, \Omega_u) \; : \; \Omega_u = \begin{bmatrix} \sigma^2_{u0} \end{bmatrix}$$
$$\text{var}(\text{cohab}_{ij}|\pi_{ij}) = \pi_{ij}(1 - \pi_{ij})/\text{denom}_{ij}$$

Now you can click on Start and the result will show in the Equations window as follows.

$$\text{cohab}_{ij} \sim \text{Binomial}(\text{denom}_{ij}, \pi_{ij})$$
$$\text{logit}(\pi_{ij}) = \beta_{0j}\text{cons}$$
$$\beta_{0j} = -0.276(0.062) + u_{0j}$$
$$\begin{bmatrix} u_{0j} \end{bmatrix} \sim \text{N}(0, \Omega_u) \; : \; \Omega_u = \begin{bmatrix} 0.032(0.023) \end{bmatrix}$$
$$\text{var}(\text{cohab}_{ij}|\pi_{ij}) = \pi_{ij}(1 - \pi_{ij})/\text{denom}_{ij}$$

This is the result presented also in Table 14.1 of Snijders & Bosker (1999).

If you also wish to reproduce the results of Example 14.3 (Table 14.2) in Snijders & Bosker (1999), you first need to obey macro IS12TRAN.MAC for some additional data manipulations. This makes available the transformed age variables used for Example 14.3. The functions defined in this example correspond to the variable names as

- $X_1(t) = age$
- $X_2(t) = ag2$
- $X_3(t) = ag2{>}10$
- $X_4(t) = ag2{>}20$.

(Age is measured in years minus 20, i.e., age is centered in this data set with 20 as the reference value.) By including these four variables with fixed effects and estimating the parameters, you will obtain the results presented as Model 1 in Table 14.2. To obtain the results of Model 2, also include a fixed effect of the variable *relig*.

8.1 Aggregation

It can be helpful to aggregate dichotomous casewise data to counts with binomial distributions in subgroups, or to relative frequencies. This will lead to smaller numbers of cases in the analysis which can be helpful for big data sets. Since the algorithms for non-linear multilevel models (of which the binomial models are a point in case) are less stable, this can also contribute to the stability of the algorithm for parameter estimation. This subsection presents an example of how this can be done in MLwiN. It is written in a way that still focuses on the Command Interface; it may be handier to use the Macro Editor instead.

The example again uses the Norwegian cohabitation data set of the ISSP 1994. Suppose that one is interested how cohabitation experience is related to religion and sex. Including the variables *sex* and *relig* and estimating the model yields the results presented in the Equations window as follows. The number of cases in the data set is 2079.

$$\text{cohab}_{ij} \sim \text{Binomial}(\text{denom}_{ij}, \pi_{ij})$$
$$\text{logit}(\pi_{ij}) = \beta_{0j}\text{cons} + 0.111(0.091)\text{sex}_{ij} + \text{-}1.800(0.223)\text{relig}_{ij}$$
$$\beta_{0j} = \text{-}0.321(0.153) + u_{0j}$$
$$\begin{bmatrix} u_{0j} \end{bmatrix} \sim \text{N}(0,\ \Omega_u) \ :\ \Omega_u = \begin{bmatrix} 0.024(0.021) \end{bmatrix}$$
$$\text{var}(\text{cohab}_{ij} | \pi_{ij}) = \pi_{ij}(1 - \pi_{ij})/\text{denom}_{ij}$$

Now the data will be aggregated per region in categories defined by sex and religion. This can be done by employing the Multilevel Data Manipulation facilities in MLwiN. Briefly, MLwiN will be tricked into treating the data as a three-level data set, with regions at level three and the sex-and-religion categories at level two; this will enable us to use the counting facility in MLwiN to create the appropriate category counts.

First a new variable is constructed that indicates the combined categories of sex and religion. This is done by calculating a new variable as follows in the command interface (check that *c13* still is an empty column):

```
calc c13 = 10*"sex"+"relig"
name c13 "sex&relig"
```

Since *sex* assumes values 1 and 2 while *relig* assumes only values 0 and 1, this recodes these variables in a combined variable with codes 10, 11, 20, 21. As a second step we need to reorder the data according to this variable. Let us make a new part of the data set. We shall need the variables *sex, reg, cons, cohab, denom, sex&relig, relig*. Go to the Data Manipulation menu and click on Sort. In the Sort window, choose 2 keys to sort on, and select for the Key code columns the variables *reg* and *sex&relig*. The first sorting key is the region variable *reg*, the second sorting key is the combined classification variable *sex&relig*. For input columns, select the seven variables *sex, reg, cons, cohab, denom, sex&relig, relig*. For output variables, select *C41* to *C47* (this is done to keep these new variables separated from the old ones). Click on Add to Action List and then click on Execute. In the Names window, you now can give new names to these seven columns; e.g., use the names *sex_a, reg_a, cons_a, cohab_a, denom_a, sex&relig_a, relig_a* where the '*_a*' stands for '*a*ggregated'. In the Data window, have a look at these data, and check that they still are ordered according to region *reg_a* and within regions they are ordered according to *sex&relig_a*. This is essential to the following aggregation operation.

The next step is the data aggregation. Define the auxiliary variable

```
calc c48 = 100*"reg_a"+"sex&relig_a"
name c48 "aux"
```

which defines the basic categories within which the data are to be aggregated. Given that *sex&relig_a* assumes no values outside the interval from 0 to 99, the regions are coded in the hundreds which is completely separated from the coding of sex and religion. In the Data Manipulation – MultiLevel data manipulations window, choose the following two transformations:

1. Operation: Count, Blocks defined by: aux, Output Column: C49 and click Add to action list;
2. Operation: Sum, Blocks defined by: aux, Input column: cohab_a, Output Column: C50 and click Add to action list.

The variables *c49* and *c50* are, respectively, the total number and the number with cohabitation experience in the combined categories of *sex* and *relig* within *reg*. For these new variables, use the names

```
name c49 "number_a" c50 "conum_a"
```

They still are replicated *number_a* times, however. This is corrected by choosing the Data Manipulation – unreplicate window, and use the following
Take specification:

Take first entry in blocks defined by: *aux*
Input columns *sex_a, reg_a, cons_a, sex&relig_a, relig_a, aux, c49, c50*
Output columns *c61–c68*

Add these to the action list and execute them. As you will see in the Names window, this generates eight variables of length 76, which is the number of non-empty categories of sex by religion within regions. It will be helpful to give these names:

```
name c61 "sex_aa" c62 "reg_aa" c63 "cons_aa" c64 "sex&relig_aa"
name c65 "relig_aa" c66 "aux_aa" c67 "number_aa" c68 "conum_aa"
```

These are the new cases and variables for the analysis. Note that the number of cases has gone down from 2079 to 76.

For the binomial analysis with the aggregated data, the relative frequencies rather than the totals *conum_aa* are needed. Therefore calculate

```
calc c69 = "conum_aa"/"number_aa"
name c69 "cofrac_aa"
```

Finally, in the Data Manipulation – MultiLevel data manipulations window, choose the transformation:

Operation: Sequence, Blocks defined by: *reg_aa*, Output Column: *C70*

and again click Add to action list and Execute. Give variable *C70* the new name *id1*, because it will be the new unit identifier at level 1. It assumes values 1 to 4, corresponding to the 4 possible combined categories of sex and religion.

This finishes the data transformation; save your worksheet if you have not already done so!

To start with a new model, the command

```
clear
```

can be given. Now in the Equations window, specify the dependent variable *cofrac_aa*, with two levels indentified by *reg_aa* and *id1*, and the binomial distribution with binomial denominator *number_aa*. Define the first explanatory variable as *cons_aa*, with a fixed effect and a random effect at level 2. Now the Equations window should look like

$$\text{cofrac\_aa}_{ij} \sim \text{Binomial}(\text{number\_aa}_{ij}, \pi_{ij})$$
$$\text{logit}(\pi_{ij}) = \beta_{0j}\text{cons\_aa}$$
$$\beta_{0j} = \beta_0 + u_{0j}$$
$$\begin{bmatrix} u_{0j} \end{bmatrix} \sim \text{N}(0,\ \Omega_u)\ :\ \Omega_u = \begin{bmatrix} \sigma^2_{u0} \end{bmatrix}$$
$$\text{var}(\text{cofrac\_aa}_{ij} | \pi_{ij}) = \pi_{ij}(1 - \pi_{ij})/\text{number\_aa}_{ij}$$

where the formula

$$\text{var}(cofrac\_aa_{ij} \mid \pi_{ij}) = \pi_{ij}(1 - \pi_{ij})/number\_aa_{ij}$$

follows from the properties of the binomial distribution.

Estimating the parameters gives the following results:

$$\text{cofrac\_aa}_{ij} \sim \text{Binomial}(\text{number\_aa}_{ij}, \pi_{ij})$$
$$\text{logit}(\pi_{ij}) = \beta_{0j}\text{cons\_aa} + 0.111(0.091)\text{sex\_aa}_{ij} + \text{-}1.800(0.223)\text{relig\_aa}_{ij}$$
$$\beta_{0j} = \text{-}0.321(0.153) + u_{0j}$$
$$\begin{bmatrix} u_{0j} \end{bmatrix} \sim \text{N}(0,\ \Omega_u)\ :\ \Omega_u = \begin{bmatrix} 0.024(0.021) \end{bmatrix}$$
$$\text{var}(\text{cofrac\_aa}_{ij} | \pi_{ij}) = \pi_{ij}(1 - \pi_{ij})/\text{number\_aa}_{ij}$$

These are exactly equal to the results obtained for the non-aggregated data. (In other cases, there can be differences in the last decimals.)

9 Some important commands

It can be a nuisance looking in the MLwiN Command Interface Manual for the most important commands – there are so many of them... Above we already encountered the following commands:

1. calc for calculation
2. logon for starting a log file
3. loga for appending a log file
4. assign for entering data (assigning values to columns)
5. echo for echoing commands to the output window and log file
6. name for naming columns
7. obey for executing macros
8. code for defining variables consisting of consecutive numbers 1 to n, possibly repeated;
9. dumm for defining dummy variables
10. mlco for defining multilevel count variables
11. take for taking out the disaggregation in higher-level variables
12. mlav for defining multilevel averages
13. fixe to show the estimates of fixed effects in the output window
14. rand to show the estimates of random part parameters in the output window
15. olse to show the estimates for an OLS (ordinary least squares) fit
16. sort for sorting (ordering) a variable
17. print for printing the values of a variable.

The commands logon and echo are *toggle commands*: e.g., if the current state of echoing is "off", then the command echo will change it to "on"; whereas if the current state is "on", then the command echo changes it to "off". There are the alternative forms echo 1 for turning on, and echo 0 for turning off irrespective of the current state. For reading more about all these commands, check the help file (e.g., by looking up command echo in the Help index) or search for the term in the MLwiN Command Manual.

Some other commands are the following. Note that the commands reporting output in the Output window will write this output to the log file, provided that the log file is being written and the Output window is open.

18. resp selects the response variable
19. expl selects explanatory ("independent") variables (watch out, this also is a toggle!)
20. iden k selects a variable to be a level identifier at level k (where k is a number)
21. setv selects a variable to have a random effect
22. clearv deletes a random effect from the model
23. start begins the estimation algorithm
24. next continues the estimation algorithm
25. batch 1 requests batch processing of estimation runs, and is a necessary command for estimating parameters (otherwise start and next will result in only one step in the estimation algorithm)
26. mlse defines a new variable containing case sequence numbers within higher-level units (this is an alternative to code which is useful also for non-balanced data)
27. ml.. there are a variety of multilevel data manipulation commands, all beginning with the letters ml

An extensive example of commands can be found in the macro file CH458.OBE, which can be used to reproduce the tables in Chapters 4, 5, and 8 of Snijders and Bosker (1999), and which is used above in Section 5.

References

Snijders, Tom A.B., and Roel J. Bosker (1999). *Multilevel Analysis. An introduction to basic and advanced multilevel modeling.* London, Thousand Oaks CA, New Delhi: Sage Publications Ltd.

Snijders, Tom A.B., and Johannes Berkhof (2007). Diagnostic checks for multilevel models. Chapter 3 in Jan de Leeuw & Erik Meijer (eds.), *Handbook of Multilevel Analysis.* Berlin, Heidelberg, New York, Hong Kong, London, Milan, Paris, Tokyo: Springer. In press. Available from `http://stat.gamma.rug.nl/snijders/publ.htm`