Example session MLwiN

*Tom A.B. Snijders**

March 2003

This is an introduction to MLwiN version 1.10.0006. The introduction is built up by guiding you through a MLwiN session. 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 user's manual (which can be downloaded from `http://multilevel.ioe.ac.uk/index.html`); but the Help facility within MLwiN may make this written documentation almost superfluous.

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 at each of them.

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 and then to An introduction to MLwiN. Have a look at this text, and then click on the link (near the top of the text) to the description of the MLwiN opening menu screen. This takes you to a screen with descriptions of the various menu items. Have a look at some of these.

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.

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

3. For an impression of the way in which more specific information is given, look at Calculation 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.

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. Macros can also be executed by using the Obey command in the Command Interface.

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.)

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 another program such as *Word* or *Wordperfect* 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, hidden to you, by the program.[2]

To request a log file choose Data Manipulation – Command Interface. You come automatically into the bottom line of the Command Interface. There type in

[1] The Command interface is very similar to the interface of the earlier DOS version MLn, but extended with extra commands.

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

```
LOGON EXA1.LOG
```

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, 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 C400 . In addition, constants can be stored in "boxes", indicated by B1 to B400 . Columns C90 through C99 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 400 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 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 is easiest if the original data set is already ordered correctly.

1.3.1 Direct input of ASCII data

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; MLwiN defines the missing value code 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.,

[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.

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

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/multilevel/`.
This is an ASCII (text) file. Have a look at the file by means of some text editor (be careful not to save it as a Word or Wordperfect or 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 screen. 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 screen 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, choose Data Manipulation – Command Interface. You come automatically into the bottom line of the Command Interface. There type in

```
OBEY MLBOOK1.DAT
```

After this command is executed, the output screen contains both the commands in the macro and the responses of MLwiN – if any – to these commands. 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.

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 the command interface.

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
```

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.

Another way is to go to the Command Interface and there issue consecutively the two commands

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

(type in the first line, hit the *Enter* key, then do the same for the next line). The code command defines a new variable C26 with all 2287 values equal to 1, which subsequently gets the name *cons*.

Instead of the code command, in the Command Interface 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 give the command

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

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 command interface, type

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

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 Data window to see how variables C14, C27, C28, and C29 hang together.

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.

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 facility[5] 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.

If you wish to analyse 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; this data set contains only the grade 8 pupils. Now the correlation (on the school level) between group size and mixed grade classes can be calculated in the Basic Statistics – Averages and Correlations window; the correlation is -0.66, confirming that small grade groups are pooled together.

[5] Corresponding to the Take command for use in the Command Interface.

1.4.4 Centering

The next steps are to subtract the averages from the variables C3 (*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.

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 at 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.

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

$$\mathbf{langpost}_{ij} = \beta_{0ij}\mathbf{cons}$$

$$\beta_{0ij} = 40.364(0.426) + u_{0j} + e_{0ij}$$

$$\begin{bmatrix} u_{0j} \end{bmatrix} \sim \mathrm{N}(0,\ \Omega_u) \ :\ \Omega_u = \begin{bmatrix} 19.419(2.921) \end{bmatrix}$$

$$\begin{bmatrix} e_{0ij} \end{bmatrix} \sim \mathrm{N}(0,\ \Omega_e) \ :\ \Omega_e = \begin{bmatrix} 64.569(1.967) \end{bmatrix}$$

$$-2*loglikelihood(IGLS) = 16253.220(2287 \text{ of } 2287 \text{ cases in use})$$

You can get this in e.g. a Word file in the following way. In MLwiN, click on the Edit window and select Copy; go to the Word file and 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 use 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. 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. This yields the results

```
N       =            2287
Average =    6.2456e-005
S.D.    =          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.

For the dependent variable *langpost* we get

```
Average =          40.935
S.D.    =          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. There appears a variable x_1 with a coefficient β_1. The fact that this coefficient does not have an extra subscript j means that it does not depend on the group, i.e., it is only a fixed and not also a random coefficient. The variable x_1 is red, showing that it is still an abstract entity and no actual variable was assigned to it.

Now click on this red variable, and a little window opens in which you can choose the variable that is to be included in the fixed part of the model. First try *iq_verb*, and when the selection is confirmed (by pressing the Done button) you will see that variable x_1 now has been changed into x_{1ij} because MLwiN detected that this is a level-one variable (it depends on the pupil, and therefore must be represented formally with both indices i and j). You can try this feature out by clicking on this variable again and changing it into *schoolse*, the mean *ses* in the school. Since this is a school variable, after confirming this choice it is represented by 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 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. 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.

$$\text{langpost}_{ij} \sim N(XB, \Omega)$$
$$\text{langpost}_{ij} = \beta_{0ij}\text{cons} + 2.488(0.070)\text{iq\_verb}_{ij}$$
$$\beta_{0ij} = 40.609(0.307) + u_{0j} + e_{0ij}$$
$$\begin{bmatrix} u_{0j} \end{bmatrix} \sim N(0, \Omega_u) : \Omega_u = \begin{bmatrix} 9.497(1.516) \end{bmatrix}$$
$$\begin{bmatrix} e_{0ij} \end{bmatrix} \sim N(0, \Omega_e) : \Omega_e = \begin{bmatrix} 42.227(1.286) \end{bmatrix}$$
$$-2*loglikelihood(IGLS) = 15251.770(2287 \text{ of } 2287 \text{ cases in use})$$

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). Also 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, 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 pop-up window for each variable in our two-level model has three boxes that could be checked:

☐ Fixed parameter
☐ j (Schoolnr)
☐ i (Pupilnr)

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.

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 changes 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 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).

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

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

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

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

$$\begin{bmatrix} u_{0j} \\ u_{1j} \end{bmatrix} \sim \text{N}(0,\ \Omega_u) \ :\ \Omega_u = \begin{bmatrix} 7.919(1.315) & \\ -0.821(0.267) & 0.200(0.098) \end{bmatrix}$$

$$\begin{bmatrix} e_{0ij} \end{bmatrix} \sim \text{N}(0,\ \Omega_e) \ :\ \Omega_e = \begin{bmatrix} 41.351(1.287) \end{bmatrix}$$

$$-2*\textit{loglikelihood(IGLS)} = 15213.530(2287 \text{ of } 2287 \text{ cases in use})$$

Note that we obtain a 2×2 covariance matrix at level 2: the random intercept variance (7.919), the slope-intercept covariance (-0.821), 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.

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 compute a cross-level interaction 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.)

The interaction variable is defined by the calculation

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

and gets the name *z2*iq*.

Add the main effect of groupsize and the interaction effect to the model, both as fixed effects only, and 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.

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

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

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

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

$$\begin{bmatrix} u_{0j} \\ u_{1j} \end{bmatrix} \sim \text{N}(0,\ \Omega_u) \ :\ \Omega_u = \begin{bmatrix} 7.666(1.283) & \\ -0.768(0.259) & 0.178(0.095) \end{bmatrix}$$

$$\begin{bmatrix} e_{0ij} \end{bmatrix} \sim \text{N}(0,\ \Omega_e) \ :\ \Omega_e = \begin{bmatrix} 41.362(1.288) \end{bmatrix}$$

*-2*loglikelihood(IGLS)* = 15208.390(2287 of 2287 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*. 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 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-C302
```

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 posterior intercepts and slopes.

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.

Of course there are many other useful commands in MLwiN for fitting more complicated models and for data manipulation. The user's manual will help you further.

5 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). 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), which will result in importing the data set with the variable labels. When you have obeyed this macro, looking at Data Manipulation – Names will show that you have a worksheet with 15 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 *three* variables all equal to 1 for

all cases. Given that the data set already contains the variable C14 = *cons*, the quickest way to achieve this is by issueing the commands

```
calc c16 = c14
calc c17 = c14
name c16 "bcons" c17 "denom"
```

The name *bcons* is shorthand for *binomial constant*, *denom* for *binomial denominator*. The first variable will be used for technical purposes only (don't think deeply about this, just accept that it is necessary), the second 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. The name *denom* is obligatory for this variable.

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 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 click on Binomial in the little window that appears.

- 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 *bcons* variable, and select only the i(respnr) field (this means that you must deselect the Fixed Parameter field!).
  In the binomial specification, this is necessary to define the level-one variance, and it comes instead of the level-one random effect for the usual constant *cons*.

- Click on the Add term button in the bottom bar of the Equations window, the on the red x_1 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* is specified.

These steps should be sufficient to define the empty model for multilevel logistic regression. Now you can click on Start and the result will show (if you have clicked once on Name and twice on Estimates in the bottom bar of the Equations window) in the Equations window as follows.

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

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

$$\beta_{1j} = -0.276(0.062) + u_{1j}$$

$$\begin{bmatrix} u_{1j} \end{bmatrix} \sim \text{N}(0,\ \Omega_u)\ :\ \Omega_u = \begin{bmatrix} 0.032(0.023) \end{bmatrix}$$

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

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

This is the result presented also in Table 14.1 of Snijders & Bosker (1999). The value 1.000 for the level-one variance parameter is only a proportionality constant, not the level-one variance itself. You do not need to report it in tables. Formally speaking, if this parameter is denoted by Ω_e like in the MLwiN Equation window, the level-one variance of the proportion of successes is $\Omega_e\,\pi_{ij}(1-\pi_{ij})/denom_{ij}$, where π_{ij} is the probability of a positive ("yes") outcome for the cases (for $denom = 1$, there is only one case) indicated by (i, j).

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 IS12TRANS.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

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

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*.