Manual supplement for MLwiN Version 2.26

Jon Rasbash
Chris Charlton
Kelvyn Jones
Rebecca Pillinger

September 2012

Manual supplement for MLwiN Version 2.26

ISBN: 978-0-903024-98-3

Printed in the United Kingdom

First Printing September 2008.

Updated for University of Bristol, October 2009.

Updated for University of Bristol, September 2012.

New Features

New features for version 2.1 include:

1. Improved model specification functionality
2. A new **Customised predictions** window for constructing and graphing model predictions
3. Basic surface plotting with rotation
4. Creation and export of model comparison tables
5. A new method for estimating autocorrelated errors in continuous time
6. Saving and retrieving of Minitab, Stata, SPSS and SAS data files
7. Saving and retrieving of MLwiN worksheets in a compressed (zipped) format
8. New data manipulation commands

Acknowledgements

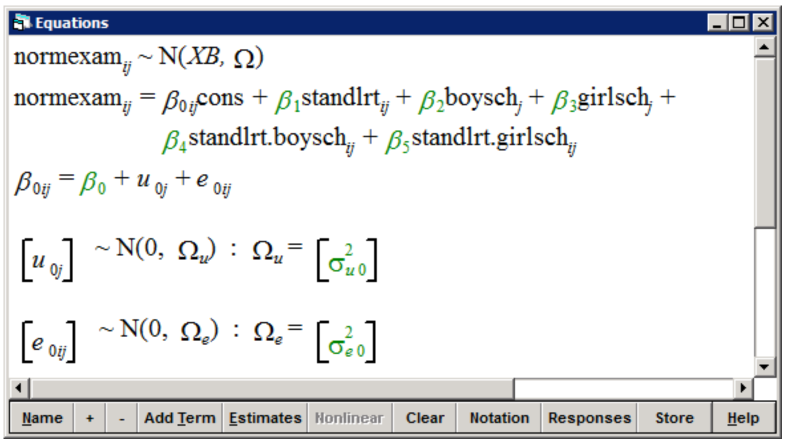
Thanks to Mike Kelly for converting this document from Word to LaTeX.

1 Improved model specification functionality

When variables and interactions were created in previous versions, recoding main effects (centring, changing reference categories) did not result in automatic recoding of the same variables wherever they appeared in interactions. Interactions and main effects had to be removed, main effects recoded and then main effects and interactions re-entered. This can be a time consuming and error prone process. In the latest version, polynomials are created by specifying an optional order. If the polynomial order, reference category (for a categorical variable) or the type of centring used for a main effect are changed then all interactions involving that variable are updated.

1.1 Recoding: reference categories, centring, and polynomials

For example, the model below, which uses the **tutorial** dataset used in chapters 1-6 of the MLwiN User's Guide, contains main effects for the continuous variable **standlrt** and the categorical variable **schgend** (consisting of dummy variables **mixedsch**, **boysch**, and **girlsch**) and the interaction of **standlrt** and **schgend**.



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

$$\text{normexam}_{ij} = \beta_{0ij}\text{cons} + \beta_1\text{standlrt}_{ij} + \beta_2\text{boysch}_j + \beta_3\text{girlsch}_j + \beta_4\text{standlrt.boysch}_{ij} + \beta_5\text{standlrt.girlsch}_{ij}$$

$$\beta_{0ij} = \beta_0 + u_{0j} + e_{0ij}$$

$$\begin{bmatrix} u_{0j} \end{bmatrix} \sim \text{N}(0,\ \Omega_u) \ : \ \Omega_u = \begin{bmatrix} \sigma^2_{u0} \end{bmatrix}$$

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

In previous versions, to change the reference category for **schgend** from **mixedsch** to **boysch** required:

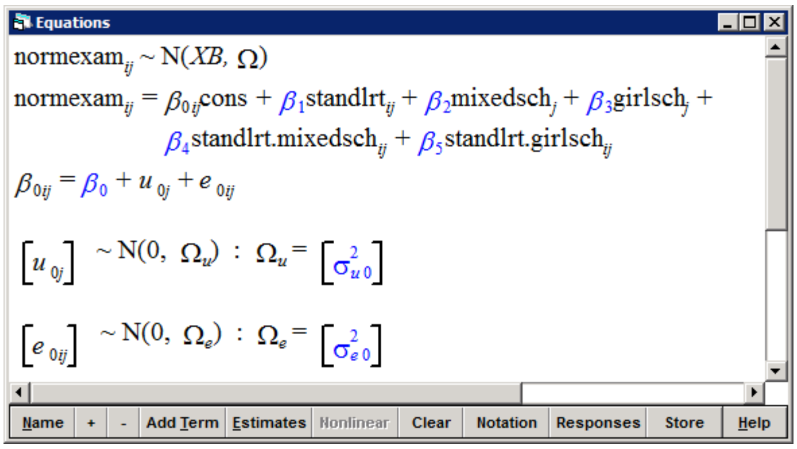
- Click on term **standlrt.boysch**
- In the window that appears, click the **delete Term** button
- When asked if the term is to be deleted, select **Yes**
- Click on term **schgend**
- In the window that appears, click the **delete Term** button
- When asked if the term is to be deleted, select **Yes**

- Click on the **Add Term** button
- Select **schgend** in the **variable** drop-down list
- Select **boysch** in the drop-down list labelled **reference category**
- Click the **Done** button
- Click on the **Add Term** button
- Select **1** from the drop-down list labelled **order**
- Select **standlrt** in the first **variable** drop-down list
- Select **schgend** in the second **variable** drop-down list
- Select **boysch** in the drop-down list labelled **reference category**
- Click the **Done** button

In version 2.1 the same operation is achieved by:

- Click on the term **boysch** (or any term that involves **schgend** categories)
- In the window that appears click the **Modify Term** button
- Select **boysch** in the drop-down list labelled **ref cat**
- Click the **Done** button

Which produces:



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

$$\text{normexam}_{ij} = \beta_{0ij}\text{cons} + \beta_1\text{standlrt}_{ij} + \beta_2\text{mixedsch}_j + \beta_3\text{girlsch}_j + \beta_4\text{standlrt.mixedsch}_{ij} + \beta_5\text{standlrt.girlsch}_{ij}$$

$$\beta_{0ij} = \beta_0 + u_{0j} + e_{0ij}$$

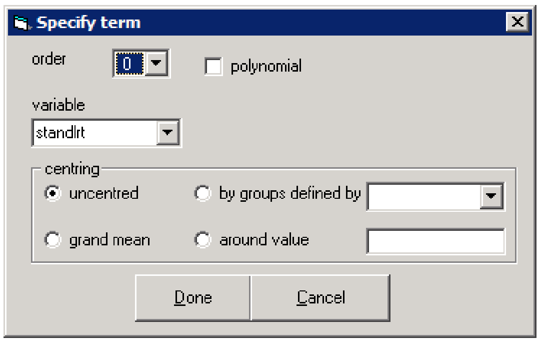
$$\begin{bmatrix} u_{0j} \end{bmatrix} \sim \text{N}(0, \Omega_u) : \Omega_u = \begin{bmatrix} \sigma^2_{u0} \end{bmatrix}$$

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

The new form of the **Specify term** window for a continuous term can be seen if you:

- Click on the term **standlrt**
- In the window that appears click the **Modify Term** button

This produces the **Specify term** window:



order: 0

polynomial: ☐

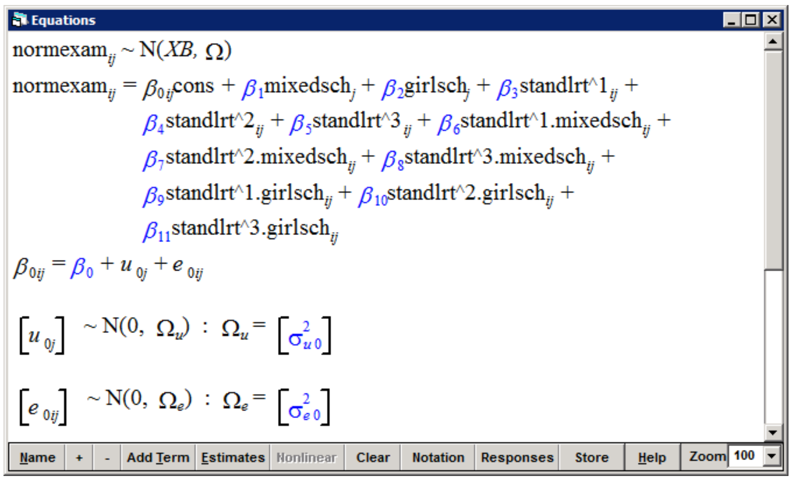
variable: standlrt

centring: [✓ uncentred] by groups defined by / grand mean / around value

If polynomial is ticked then a drop-down list appears for the degree of the polynomial. Continuous variables can be **uncentred**, centred around means of **groups defined by** codes contained in a specified column, centred around the **grand mean** (i.e. the overall mean) or centred **around value**. Thus,

- Click on the **polynomial** check box
- A drop-down list labelled **poly degree** will appear; select **3**
- Click the **Done** button in the **Specify term** window

produces:



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

$$\begin{aligned}\text{normexam}_{ij} = {} & \beta_{0ij}\text{cons} + \beta_1\text{mixedsch}_j + \beta_2\text{girlsch}_j + \beta_3\text{standlrt\^{}1}_{ij} + \\ & \beta_4\text{standlrt\^{}2}_{ij} + \beta_5\text{standlrt\^{}3}_{ij} + \beta_6\text{standlrt\^{}1.mixedsch}_{ij} + \\ & \beta_7\text{standlrt\^{}2.mixedsch}_{ij} + \beta_8\text{standlrt\^{}3.mixedsch}_{ij} + \\ & \beta_9\text{standlrt\^{}1.girlsch}_{ij} + \beta_{10}\text{standlrt\^{}2.girlsch}_{ij} + \\ & \beta_{11}\text{standlrt\^{}3.girlsch}_{ij}\end{aligned}$$

$$\beta_{0ij} = \beta_0 + u_{0j} + e_{0ij}$$

$$\begin{bmatrix} u_{0j} \end{bmatrix} \sim \mathrm{N}(0,\ \Omega_u)\ :\ \Omega_u = \begin{bmatrix} \sigma^2_{u0} \end{bmatrix}$$

$$\begin{bmatrix} e_{0ij} \end{bmatrix} \sim \mathrm{N}(0,\ \Omega_e)\ :\ \Omega_e = \begin{bmatrix} \sigma^2_{e0} \end{bmatrix}$$

Note that changing **standlrt** to be a cubic polynomial also updates the interaction of **standlrt** and **schgend** to be cubic with respect to **standlrt**. If you want a cubic main effect for **standlrt** but interaction terms with **schgend** to be linear with respect to **standlrt** then click on any of the interaction terms and set the degree of polynomial for **standlrt** to be 1.

1.2 Orthogonal Polynomials

1.2.1 Orthogonal Polynomials

Orthogonal polynomials are useful when fitting variables measured on an ordinal scale as predictors in models. An example would be Age with categories 21–30, 31–40, 41–50 and on. Instead of fitting the ordinal variable as a constant and a contrasted set of dummies (with one left out), we fit it as an orthogonal polynomial of degree at most one less than the number of categories. We thus have at most the same number of variables making up the orthogonal polynomial as we would have dummies if we fitted the ordinal variable using that specification. Unlike the dummies, the variables comprising the terms of the orthogonal polynomial are not (0,1) variables. They contain values between -1 and 1. For each term, there is a different value for each category of the ordinal variable. These values depend only on the number of categories the ordinal variable has. The values for ordinal variables with 3, 4 and 6 categories are shown below. The values give the terms of the polynomial certain properties when the data are balanced:

- They are orthogonal. In mathematical terms this means that if you pick any two terms of the polynomial, and for each category multiply the value for that category for the first term by the value for that category for the second term, then add these products together, the result will be 0: for example, picking the two terms of the polynomial for a 3 category variable, $-0.707 \times 0.408 + 0 \times -0.816 + 0.707 \times 0.408 = 0$. In statistical terms, this means that each pair of terms is uncorrelated, which turns out to be useful in modelling as we will see later: in particular estimates associated with these orthogonal variables are likely to be numerically stable.

- They each have the same mean and variance. The mean is always 0 but the variance depends on the number of categories in the ordinal variable. Again this will turn out to be useful in modelling

- Each term is a function of the appropriate power of some value. In other words, the linear term is always a linear function of some value, the quadratic term is a quadratic function of some value, the cubic term is a cubic function of some value and so on. Consequently, when an intercept and the full set of terms are included in the model, they can completely capture the effects of all the categories of the ordinal variable on the response, no matter what those effects are. Another consequence of this property is that in many cases we can achieve a more parsimonious model by using a subset of the terms, as we will see. The linear term captures the linear effect across categories of the ordinal variable, the quadratic term captures the quadratic effect, and so on. We cannot of course achieve this if we use an intercept plus a set of contrasted dummies: to leave a dummy out means conflating that category with the reference category.

Note that the current implementation in MLwiN assumes that the categories of the ordinal variable are equally spaced.

Below are three examples of coding when 3, 4 and 6 categories of an ordered variable are included (it is presumed that the model contains a constant).

Categorical Variables Codings for 3 categories

		Parameter coding	
		Linear	Quadratic
3 groups	1	-.707	.408
	2	.000	-.816
	3	.707	.408

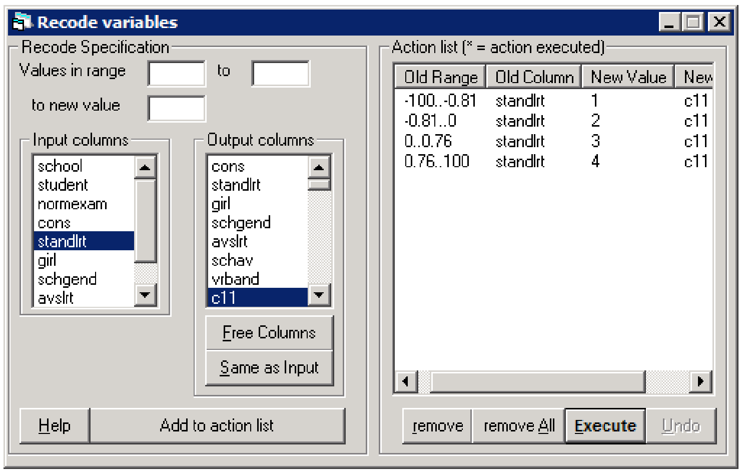
Categorical Variables Codings for 4 categories

		Parameter coding		
		Linear	Quadratic	Cubic
4 groups	1	-.671	.500	-.224
	2	-.224	-.500	.671
	3	.224	-.500	-.671
	4	.671	.500	.224

Categorical Variables Codings for 6 categories

		Parameter coding				
		Linear	Quadratic	Cubic	4^{th} order	5^{th} order
6 groups	1	-.598	.546	-.373	.189	-.063
	2	-.359	-.109	.522	-.567	.315
	3	-.120	-.436	.298	.378	-.630
	4	.120	-.436	-.298	.378	.630
	5	.359	-.109	-.522	-.567	-.315
	6	.598	.546	.373	.189	.063

As an example we will work with the **tutorial** dataset and turn **standlrt** into 4 ordered categories: this requires 3 cut points; we use the nested means procedure explained in Section 2.1.4 to get our values. Note that we are making **standlrt** into an ordinal variable purely to show the operation of the orthogonal polynomial feature; if we were really performing some analysis using a continuous variable like **standlrt** this would not be a recommended procedure. The nested means when rounded are -0.81, 0.00, and 0.76 and we use these values to recode the data into 4 groups via the **recode** $\rightarrow$ **by range** option from the **Data manipulation** menu:

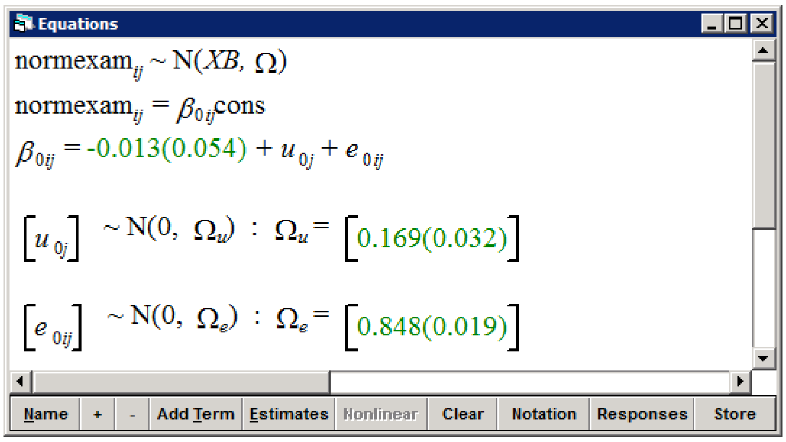


Old Range	Old Column	New Value	New
-100..-0.81	standlrt	1	c11
-0.81..0	standlrt	2	c11
0..0.76	standlrt	3	c11
0.76..100	standlrt	4	c11

- In the **Names** window, change the name of **c11** to be **LRTgrps**
- Click on **Toggle categorical** to change it from a continuous to an ordinal variable (note that the orthogonal polynomial facility can only be accessed when the variable has been predefined as categorical)

For purposes of comparison we begin with the 4 groups included in the model in the usual way for categorical variables i.e. as a constant and three contrasted dummies. We will then go on to see whether we can fit a more parsimonious model using orthogonal polynomials.

Start by setting up the model as follows:



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

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

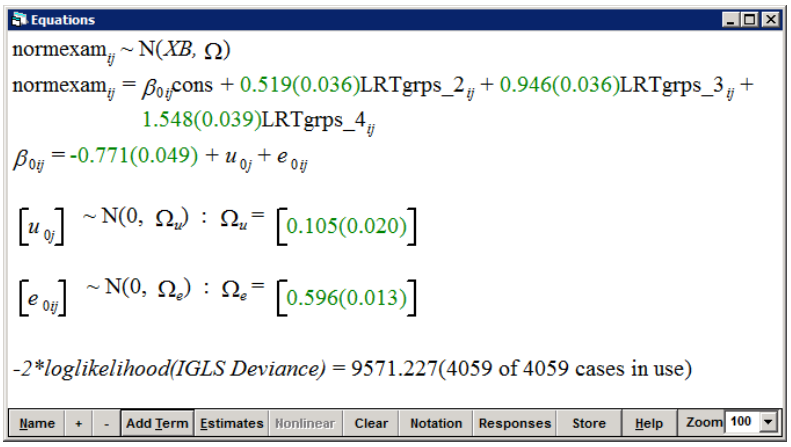
$$\beta_{0ij} = -0.013(0.054) + u_{0j} + e_{0ij}$$

$$\begin{bmatrix} u_{0j} \end{bmatrix} \sim \mathrm{N}(0, \Omega_u) : \Omega_u = \begin{bmatrix} 0.169(0.032) \end{bmatrix}$$

$$\begin{bmatrix} e_{0ij} \end{bmatrix} \sim \mathrm{N}(0, \Omega_e) : \Omega_e = \begin{bmatrix} 0.848(0.019) \end{bmatrix}$$

- Click on **Add Term**
- From the **variable** drop-down list select **LRTgrps**

- From the **ref cat** drop-down list select **LRTgrps_1**
- Click **Done**



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

$\text{normexam}_{ij} = \beta_{0ij}\text{cons} + 0.519(0.036)\text{LRTgrps\_2}_{ij} + 0.946(0.036)\text{LRTgrps\_3}_{ij} + 1.548(0.039)\text{LRTgrps\_4}_{ij}$

$\beta_{0ij} = -0.771(0.049) + u_{0j} + e_{0ij}$

$\left[u_{0j}\right] \sim \text{N}(0,\ \Omega_u) \ :\ \Omega_u = \left[0.105(0.020)\right]$

$\left[e_{0ij}\right] \sim \text{N}(0,\ \Omega_e) \ :\ \Omega_e = \left[0.596(0.013)\right]$

*-2*loglikelihood(IGLS Deviance)* = 9571.227(4059 of 4059 cases in use)

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

We now fit a model with **LRTgrps** entered as an orthogonal polynomial instead of as three dummies. We start by including all three terms of the orthogonal polynomial. First, though, we need to change the tolerance used by MLwiN. The tolerance is set by default to a value which is most appropriate for the most commonly used kinds of estimation. However, using this value of the tolerance when working with orthogonal polynomials can lead to the wrong values being calculated for the terms of the polynomial, which in turn leads to incorrect estimates for the coefficients. This is particularly true when the categorical variable has a larger number of categories. To change the tolerance,

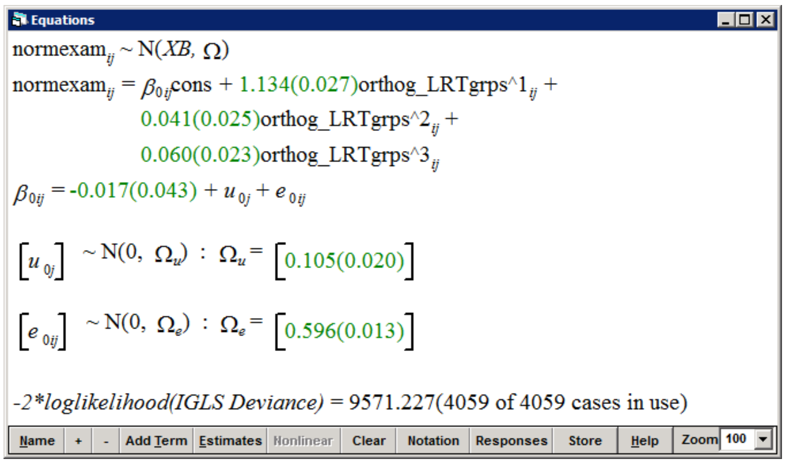
- From the **Data Manipulation** window, select **Command interface**
- In the Command interface window, type **CTOL 10**

The tolerance is now set to a value of 10^{10}. This should be sufficient in most cases. The default tolerance that MLwiN normally uses is 10^{6} and the tolerance can be set back to this value by typing `CTOL 6` in the **Command interface** window. This should be done after working with orthogonal polynomials as using a tolerance of 10^{10} can cause problems in other areas.

We can now proceed to set up the model

- Click on any of the three **LRTgrps** dummies
- Click on the **Delete Term** button
- Click **Yes**
- Click on the **Add Term** button

- Select **LRTgrps** from the **variable** drop-down list
- Tick the **orthogonal polynomial** box
- From the **orthog poly degree** drop-down list select **3** (note that the greatest degree that the software will allow you to specify is always one less than the number of categories)
- Click **Done**



Equations

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

$$\begin{aligned}\text{normexam}_{ij} = \beta_{0ij}\text{cons} &+ 1.134(0.027)\text{orthog\_LRTgrps\^{}1}_{ij} + \\ &0.041(0.025)\text{orthog\_LRTgrps\^{}2}_{ij} + \\ &0.060(0.023)\text{orthog\_LRTgrps\^{}3}_{ij}\end{aligned}$$

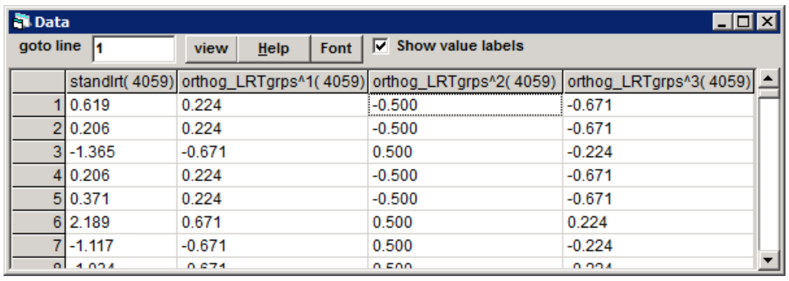
$$\beta_{0ij} = -0.017(0.043) + u_{0j} + e_{0ij}$$

$$\left[u_{0j}\right] \sim \text{N}(0,\ \Omega_u) \ : \ \Omega_u = \left[0.105(0.020)\right]$$

$$\left[e_{0ij}\right] \sim \text{N}(0,\ \Omega_e) \ : \ \Omega_e = \left[0.596(0.013)\right]$$

*-2*loglikelihood(IGLS Deviance)* = 9571.227(4059 of 4059 cases in use)

Three new variables containing the appropriate values are automatically generated as appropriately named columns in the worksheet and added to the model:



Data

goto line 1 · Show value labels

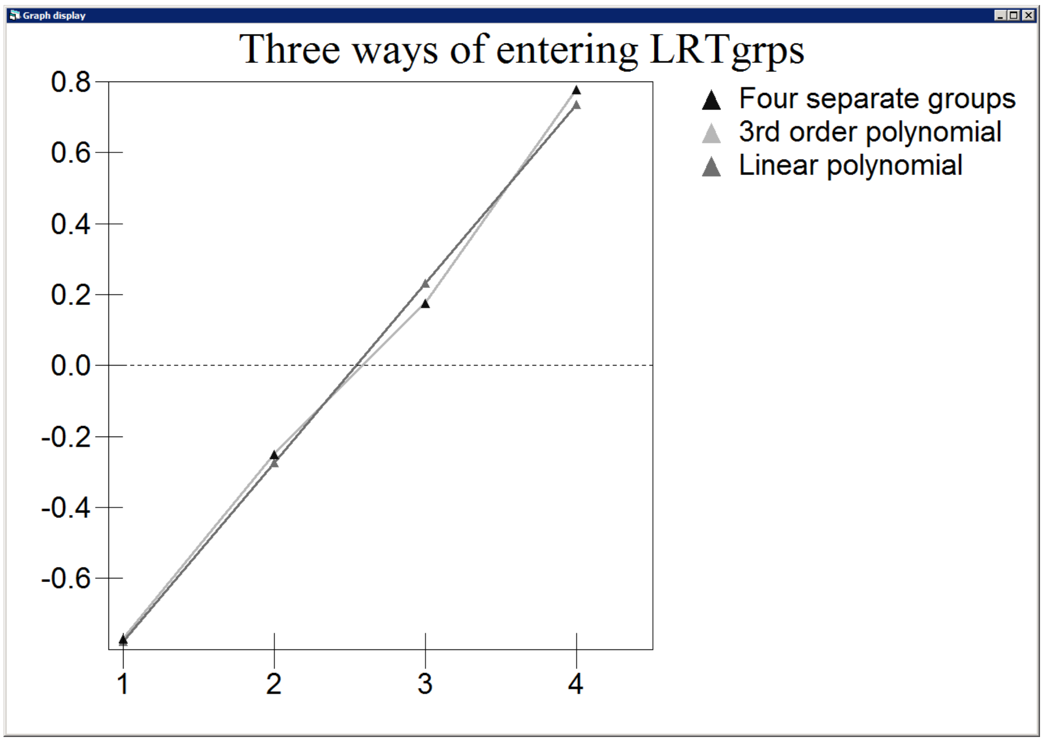
	standlrt(4059)	orthog_LRTgrps^1(4059)	orthog_LRTgrps^2(4059)	orthog_LRTgrps^3(4059)
1	0.619	0.224	-0.500	-0.671
2	0.206	0.224	-0.500	-0.671
3	-1.365	-0.671	0.500	-0.224
4	0.206	0.224	-0.500	-0.671
5	0.371	0.224	-0.500	-0.671
6	2.189	0.671	0.500	0.224
7	-1.117	-0.671	0.500	-0.224

At this point the model is equivalent to our first model with **LRTgrps** entered as dummies: it has the same deviance and the estimates for the random part are identical, although the coefficients are different because the terms of the polynomial are measured on a different scale to the dummies and the intercept now has a different value because it is the average value of the response when all three terms of the polynomial are 0, not the average value of the response for the reference category of **LRTgrps**. We can see in the graph below that these two models give us the same results.

The important difference is that the four group model is not readily reducible whereas the orthogonal polynomial one is. The three terms all have the same mean and variance so their coefficients are directly comparable and so it is clear from the estimates that the linear effect is markedly bigger than the quadratic and cubic, thereby suggesting that the model be simplified to have just a linear trend across all four ordered categories. We fit a model using just a linear effect:

- Click on any of the terms of the orthogonal polynomial
- Click **Modify Term**
- Select **1** from the **orthog poly degree** drop-down list
- Click **Done**

The graph shows that the linear trend model well captures the underlying trend in a parsimonious way. We have thus succeeded in simplifying our model without losing much information.



1.2.2 Using orthogonal polynomials with repeated measures data

Orthogonal polynomials are especially useful with repeated measures data and fitting a more parsimonious model is not the only reason to use them. Hedeker and Gibbons (2006)(3) give the following reasons to use orthogonal polynomials in their discussion of growth curves for longitudinal data where the predictor representing time is not a continuous variable but 1, 2, 3 etc representing the first, second and third occasion on which the person has been measured:

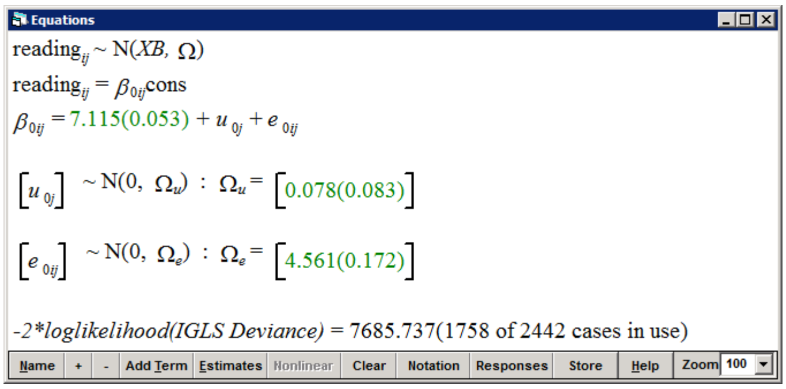
- for balanced data, and compound symmetry structure (that is a variance components model), estimates of polynomial fixed effects (e.g.

constant and linear) do not change when higher-order polynomial terms (e.g. quadratic and cubic) are added to the model

- using the original scale, it gets increasingly difficult to estimate higher-degree non-orthogonal polynomial terms
- using orthogonal polynomials avoids high correlation between estimates (which can cause estimation problems)
- using orthogonal polynomials provides comparison of the 'importance' of different polynomials, as the new terms are put on the same standardized (unit) scale. This holds exactly when the number of observations at each timepoint are equal, and approximately so when they are unequal.
- the intercept (and intercept-related parameters) represents the grand mean of the response, that is, the model intercept represents the mean of the response at the midpoint of time.

Another reason for using them is the orthogonality property: since each pair of terms in the polynomial are orthogonal, i.e. uncorrelated, the coefficients of the terms do not change according to which other terms are included. This means that if we decide that the linear and quadratic effects are the most important based on the model using all terms and go on to fit a model including just the linear and quadratic terms, we will not find for example that now the higher order terms have been removed, the quadratic effect is no longer important.

As an example of using orthogonal polynomials with repeated measures data, we will use the **reading1** dataset which forms the basis for analysis in Chapter 13 of the User's Guide. We begin with the variance components model of page 184 (you will need to follow the instructions on pages 197 to 200 and 202 to get the data in the correct form to set this model up and create a constant column):



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

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

$\beta_{0ij} = 7.115(0.053) + u_{0j} + e_{0ij}$

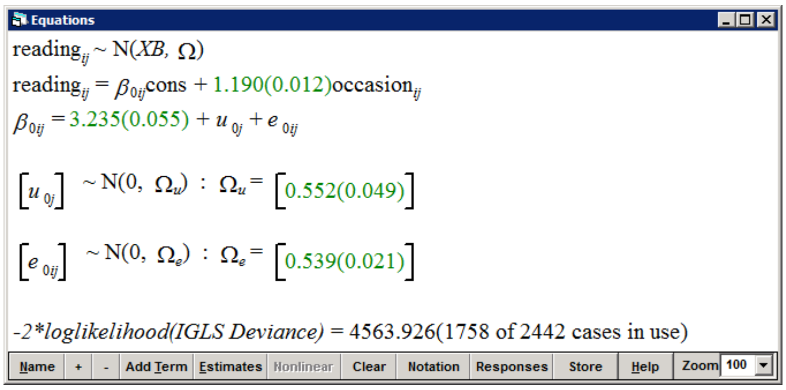
$\left[u_{0j}\right] \sim \text{N}(0, \Omega_u) : \Omega_u = \left[0.078(0.083)\right]$

$\left[e_{0ij}\right] \sim \text{N}(0, \Omega_e) : \Omega_e = \left[4.561(0.172)\right]$

*-2*loglikelihood(IGLS Deviance)* = 7685.737(1758 of 2442 cases in use)

We are going to pretend that we do not know the age at which the reading score was evaluated but only know that the reading was taken on 1st, 2nd, 3rd etc occasion. We can fit a linear trend by putting 'occasion' into the fixed part of the model:

- In the **Names** window, highlight **occasion** and click **Toggle categorical**
- Add **occasion** to the model



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

$\text{reading}_{ij} = \beta_{0ij}\text{cons} + 1.190(0.012)\text{occasion}_{ij}$

$\beta_{0ij} = 3.235(0.055) + u_{0j} + e_{0ij}$

$\left[u_{0j}\right] \sim N(0, \Omega_u) : \Omega_u = \left[0.552(0.049)\right]$

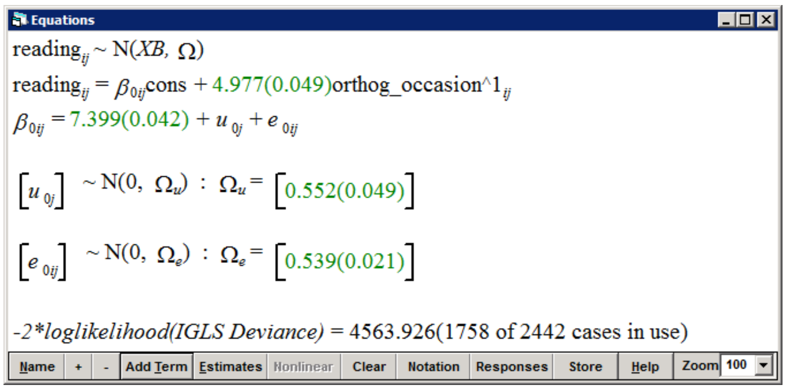
$\left[e_{0ij}\right] \sim N(0, \Omega_e) : \Omega_e = \left[0.539(0.021)\right]$

*-2*loglikelihood(IGLS Deviance)* = 4563.926(1758 of 2442 cases in use)

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

The estimate 3.235 gives the mean Reading score at occasion 0 (that is the occasion before the first measurement!) and the mean reading score improves by 1.19 for each subsequent occasion. We can now fit a similar model but replacing **occasion** by the linear orthogonal variable.

- Click on **occasion**
- Click **Delete Term**
- In the **Names** window, highlight **occasion** and click **Toggle Categorical**
- In the **Equations** window, click **Add Term**
- From the **variable** drop-down list select **occasion**
- Tick the **orthogonal polynomial** box
- From the **orthog poly degree** drop-down list select **1**
- Click **Done**



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

$\text{reading}_{ij} = \beta_{0ij}\text{cons} + 4.977(0.049)\text{orthog\_occasion\^{}1}_{ij}$

$\beta_{0ij} = 7.399(0.042) + u_{0j} + e_{0ij}$

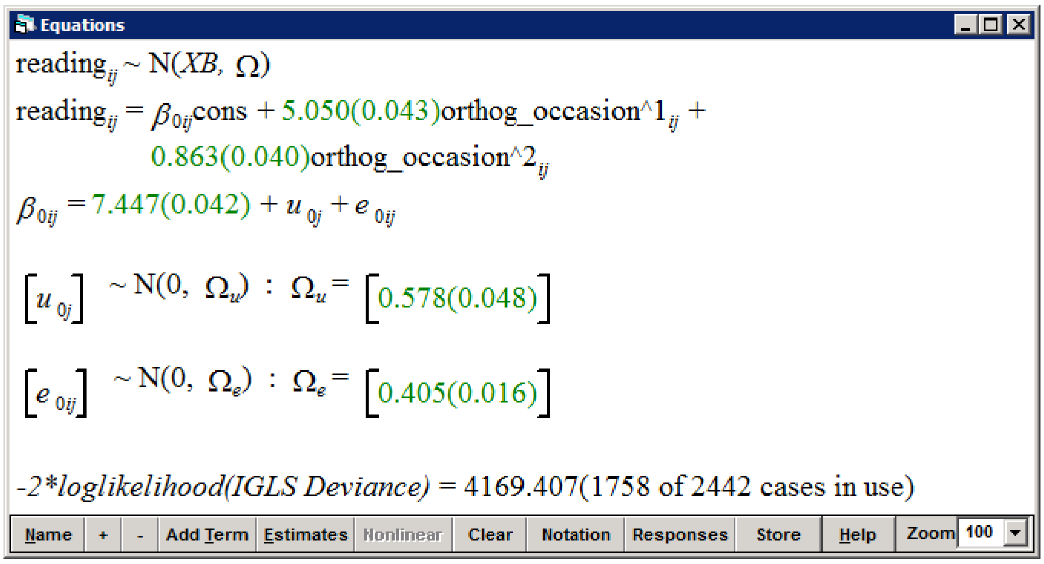
$\left[u_{0j}\right] \sim N(0, \Omega_u) : \Omega_u = \left[0.552(0.049)\right]$

$\left[e_{0ij}\right] \sim N(0, \Omega_e) : \Omega_e = \left[0.539(0.021)\right]$

*-2*loglikelihood(IGLS Deviance)* = 4563.926(1758 of 2442 cases in use)

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

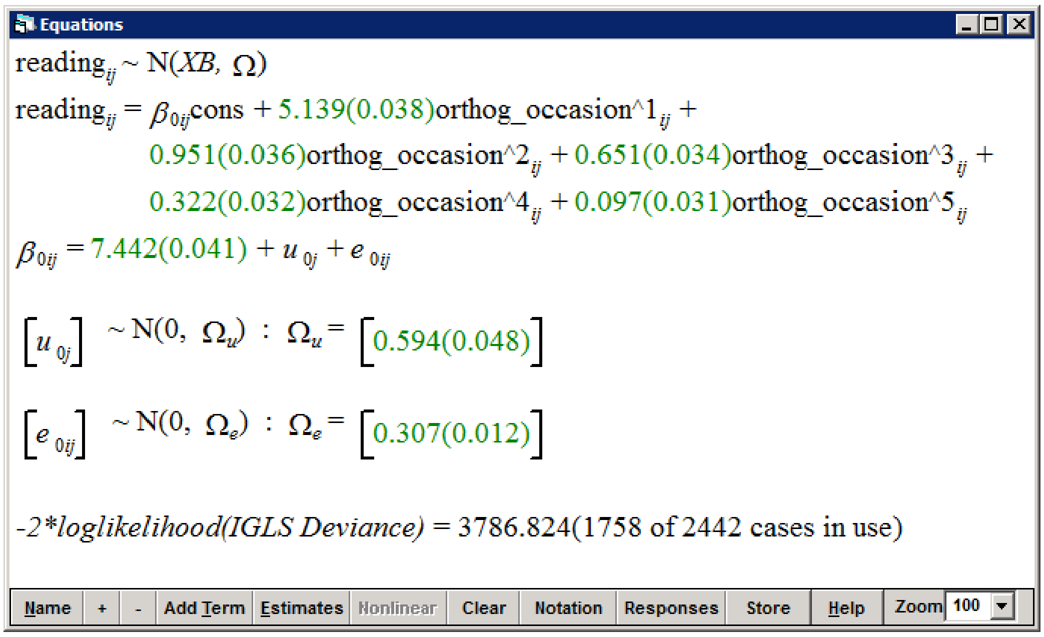
The deviance and the random part are the same (as is the Wald test for the growth term), but now the estimate of the intercept gives the grand mean at the mid-point of the occasions and the slope gives the improvement in reading with a unit change in the linear orthogonal polynomial. We now add the quadratic orthogonal polynomial into the model using the **Modify Term** option and should find that the linear term does not change (in fact it does a bit due to imbalance in the data). It is clear that the quadratic term is of much less 'importance' than the linear term.



$$\text{reading}_{ij} \sim \text{N}(XB, \Omega)$$
$$\text{reading}_{ij} = \beta_{0ij}\text{cons} + 5.050(0.043)\text{orthog\_occasion\^{}1}_{ij} + 0.863(0.040)\text{orthog\_occasion\^{}2}_{ij}$$
$$\beta_{0ij} = 7.447(0.042) + u_{0j} + e_{0ij}$$
$$\left[u_{0j}\right] \sim \text{N}(0, \Omega_u) : \Omega_u = \left[0.578(0.048)\right]$$
$$\left[e_{0ij}\right] \sim \text{N}(0, \Omega_e) : \Omega_e = \left[0.405(0.016)\right]$$

*-2*loglikelihood(IGLS Deviance)* = 4169.407(1758 of 2442 cases in use)

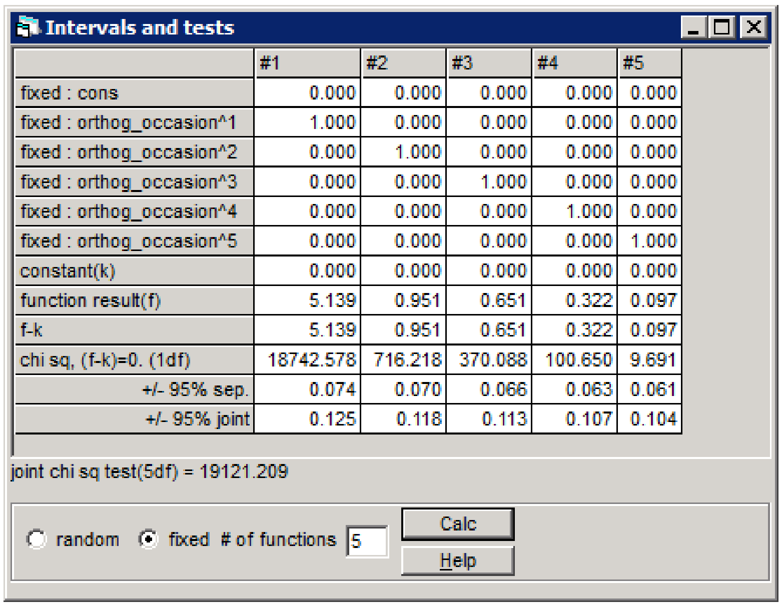
Using the **Predictions** and **Customised graphs** windows we can see the quadratic growth curve which is characterized by a strong linear trend, with some slight curvature. We can now add in all the other polynomials (there is then a term for each occasion) and it is clear that there are 'diminishing returns' as each additional order is included



$$\text{reading}_{ij} \sim \text{N}(XB, \Omega)$$
$$\text{reading}_{ij} = \beta_{0ij}\text{cons} + 5.139(0.038)\text{orthog\_occasion\^{}1}_{ij} + 0.951(0.036)\text{orthog\_occasion\^{}2}_{ij} + 0.651(0.034)\text{orthog\_occasion\^{}3}_{ij} + 0.322(0.032)\text{orthog\_occasion\^{}4}_{ij} + 0.097(0.031)\text{orthog\_occasion\^{}5}_{ij}$$
$$\beta_{0ij} = 7.442(0.041) + u_{0j} + e_{0ij}$$
$$\left[u_{0j}\right] \sim \text{N}(0, \Omega_u) : \Omega_u = \left[0.594(0.048)\right]$$
$$\left[e_{0ij}\right] \sim \text{N}(0, \Omega_e) : \Omega_e = \left[0.307(0.012)\right]$$

*-2*loglikelihood(IGLS Deviance)* = 3786.824(1758 of 2442 cases in use)

but all terms have some effect: with even the 5th order term having a p value of 0.002

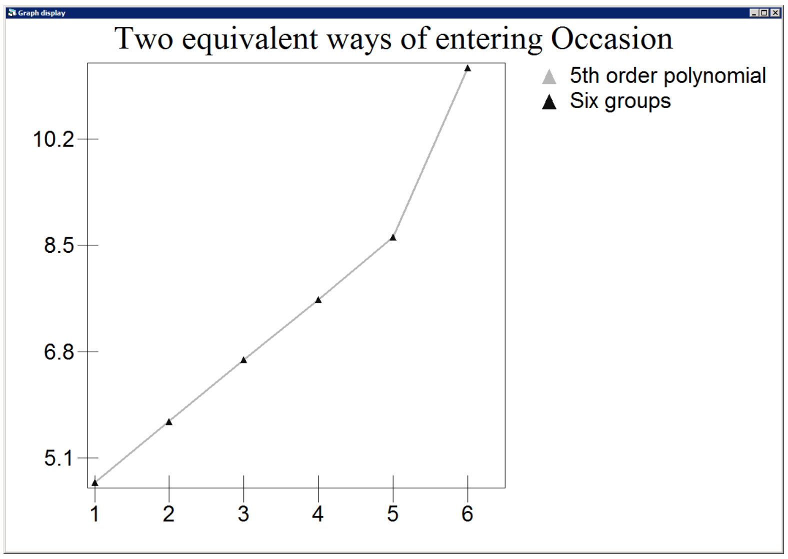


	#1	#2	#3	#4	#5
fixed : cons	0.000	0.000	0.000	0.000	0.000
fixed : orthog_occasion^1	1.000	0.000	0.000	0.000	0.000
fixed : orthog_occasion^2	0.000	1.000	0.000	0.000	0.000
fixed : orthog_occasion^3	0.000	0.000	1.000	0.000	0.000
fixed : orthog_occasion^4	0.000	0.000	0.000	1.000	0.000
fixed : orthog_occasion^5	0.000	0.000	0.000	0.000	1.000
constant(k)	0.000	0.000	0.000	0.000	0.000
function result(f)	5.139	0.951	0.651	0.322	0.097
f-k	5.139	0.951	0.651	0.322	0.097
chi sq, (f-k)=0. (1df)	18742.578	716.218	370.088	100.650	9.691
+/- 95% sep.	0.074	0.070	0.066	0.063	0.061
+/- 95% joint	0.125	0.118	0.113	0.107	0.104

joint chi sq test(5df) = 19121.209

random [✓ fixed] # of functions 5

As in the example using the tutorial dataset, this model which fits an intercept and 5 growth terms is equivalent to fitting a separate parameter for each occasion; the graph below makes this clear. However we can readily leave out one or more terms of the orthogonal polynomial to achieve a more parsimonious model; the same is not true of the intercept and dummy variables specification.



(Note that the way the response has been constructed will have a great effect on these results; see page 178 of the User's Guide)

The tolerance should now be set back to the default value before working further in MLwiN

- In the **Command interface** window, type `CTOL 6`

1.3 Commands

The ADDT command has been extended to allow specification of polynomial terms:

`ADDT` **C** <mode **N**> **C** <mode **N**>

Create a main effect or interaction term from a series of one or more variables. Each variable can be categorical or continuous. Categorical variables can have a reference category specified, by setting the corresponding mode value to the number of the reference category. If no reference category is specified, then the lowest category number is taken as the reference category. If -1 is given as the reference category then no reference category is assumed and a full set of dummies are produced. Variables of appropriate names and data patterns are created and added as explanatory variables. If $N > 10000$, then $N - 10000$ is taken to be the polynomial degree that is required for the corresponding variable. If the corresponding variable is categorical then an orthogonal polynomial of degree $N - 10000$ is fitted.

Examples:

```
► ADDT 'standlrt' 10003 'schgend'
```

adds an interaction between a cubic in **standlrt** and 2 dummies for **schgend** with reference category **mixedsch.**

```
► ADDT 'standlrt' 'schgend' 10002
```

adds an interaction term between 'standlrt' and an orthogonal polynomial of degree 2 for **schgend**.

The SWAP command edits a main effect and updates all affected interactions

SWAP main effect **C** with **C** mode **N**

Mode N has the same meaning as in the ADDT command.

Examples:

```
▶ SWAP 'standlrt' 'standlrt' 10003
```

removes **standlrt** as a main effect and all interactions involving **standlrt** from the model and replaces them with a cubic polynomial in **standlrt**; note all interactions involving **standlrt** are also replaced.

Note that the SWAP command will not do anything if the variable you ask to swap is only present in an interaction term.

The CENT command controls centring for continuous explanatory variables

CENT mode **0**:	uncentred
CENT mode **1**:	around grand mean
CENT mode **2**:	around means of groups defined by codes in **C**
CENT mode **3**:	around value **N**

Examples:

```
▶ CENT 2 'school'
▶ ADDT 'standlrt'
```

adds the variable **standlrt** to the model centered around the group mean of **standlrt**, where groups are defined by the codes in the column ‘school’

```
▶ CENT 0
▶ ADDT ‘standlrt'
```

adds **standlrt** to the model with no centring.

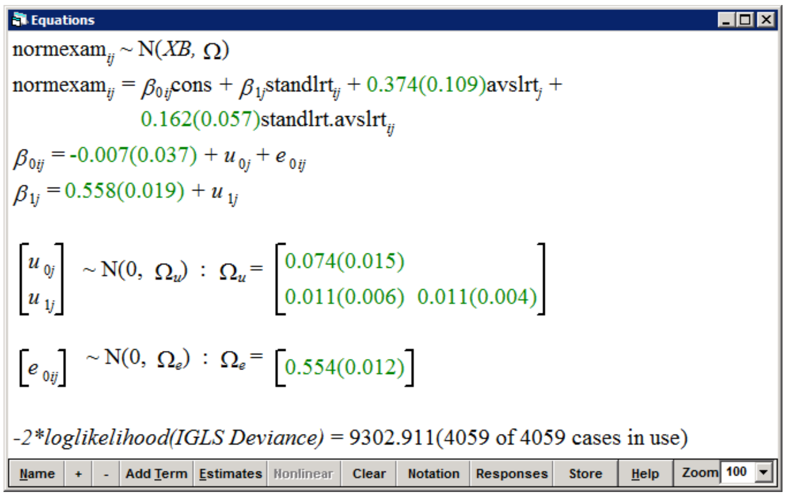
The default for CENT is no centring: if ADDT is used without CENT having been previously used, then the term is added with no centring. The kind of centring set up by use of the CENT command remains in place until the CENT command is used again to change it: it is not necessary to use the CENT command before every ADDT if the same kind of centring is required in each case.

2 Out of Sample predictions

2.1 Continuous responses

The current **Predictions** window generates a predicted value of the response for each level 1 unit in the dataset, using their values of the explanatory variables. Thus, predictions are only generated for the combinations of values of the explanatory variables occurring in the dataset. Often, however, we want to generate predictions for a specific set of explanatory variable values to best explore a multidimensional model predictions space.

We have added a **Customised Predictions** window, with an associated plot window, to aid this task by allowing predictions to be generated for any desired combinations of explanatory variable values, whether or not these combinations occur in the dataset. Given the model on the **tutorial** data



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

$$\text{normexam}_{ij} = \beta_{0ij}\text{cons} + \beta_{1j}\text{standlrt}_{ij} + 0.374(0.109)\text{avslrt}_j + 0.162(0.057)\text{standlrt.avslrt}_{ij}$$

$$\beta_{0ij} = -0.007(0.037) + u_{0j} + e_{0ij}$$

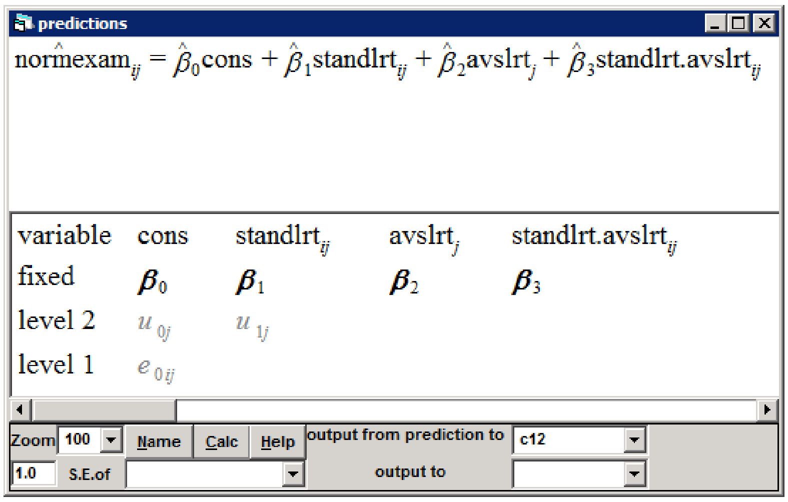
$$\beta_{1j} = 0.558(0.019) + u_{1j}$$

$$\begin{bmatrix} u_{0j} \\ u_{1j} \end{bmatrix} \sim \text{N}(0,\ \Omega_u)\ :\ \Omega_u = \begin{bmatrix} 0.074(0.015) & \\ 0.011(0.006) & 0.011(0.004) \end{bmatrix}$$

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

*-2*loglikelihood(IGLS Deviance)* = 9302.911(4059 of 4059 cases in use)

where **avslrt** is the school mean for **standlrt**, we can use the old **Predictions** window to explore this relationship



$$\hat{\text{normexam}}_{ij} = \hat{\beta}_0\text{cons} + \hat{\beta}_1\text{standlrt}_{ij} + \hat{\beta}_2\text{avslrt}_j + \hat{\beta}_3\text{standlrt.avslrt}_{ij}$$

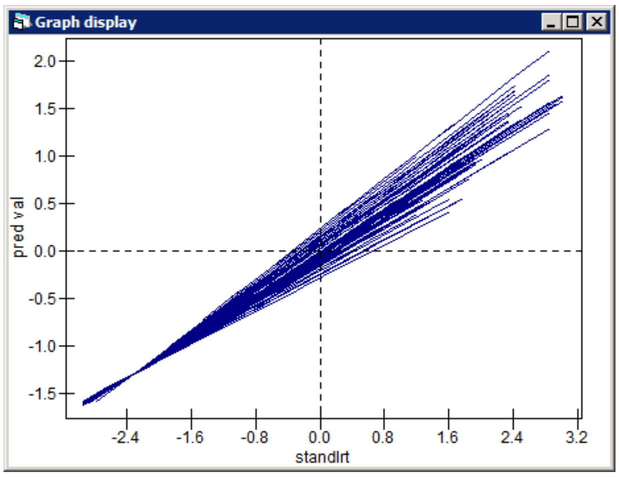
variable	cons	standlrt_{ij}	avslrt_j	$\text{standlrt.avslrt}_{ij}$
fixed	β_0	β_1	β_2	β_3
level 2	u_{0j}	u_{1j}		
level 1	e_{0ij}			

output from prediction to c12

1.0 S.E.of

output to

This window applies the specified prediction function to every point in the dataset and produces a predicted value for each point. If we plot these predictions, grouped by school, we get

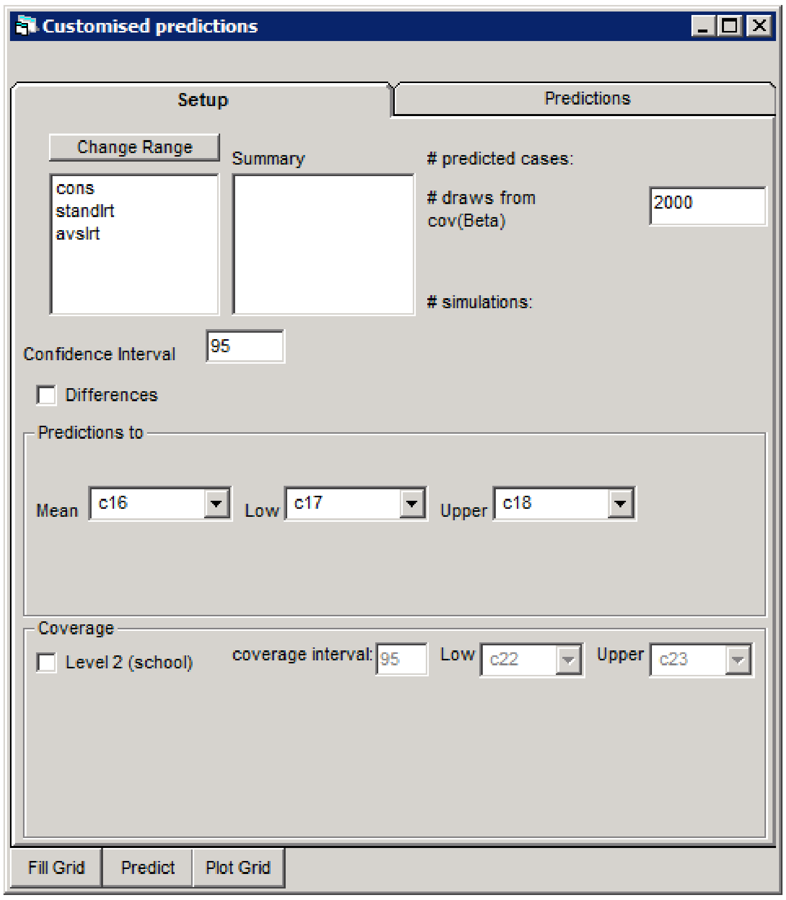


This plot has one line for each school; the line for the jth school is given by the prediction equation $(\hat{\beta}_0 + \hat{\beta}_2\mathbf{avslrt}_j) + (\hat{\beta}_1 + \hat{\beta}_3\mathbf{avslrt}_j)\mathbf{standlrt}_{ij}$. However, it might be more revealing to plot lines for a small number of different values of **avslrt,** say {-0.5, 0, 0.5}

The **Customised predictions** window will now do this and other such prediction tasks automatically for us.

- Select **Customised Predictions** from the Model menu.

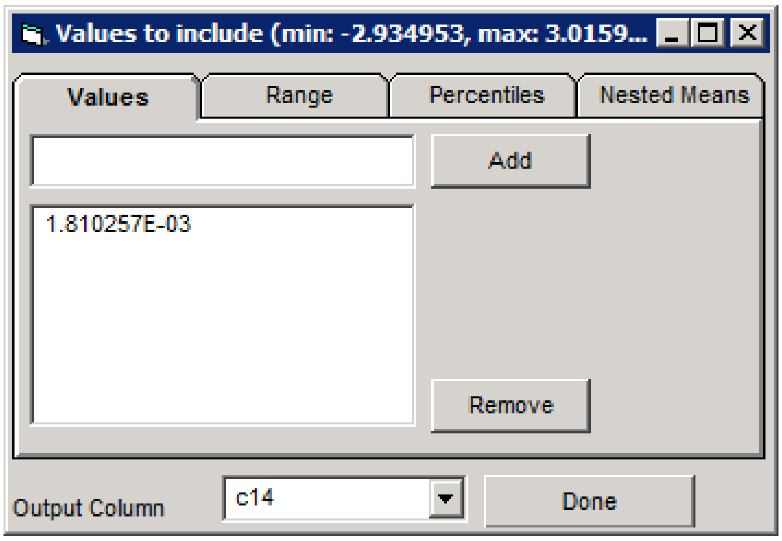
The following screen appears:



Main effects are listed in the setup pane. We can click on them and specify a set of values we want to make a prediction dataset for.

- Highlight **standlrt**
- Click **Change Range**

We see:



Currently the value is set to the mean of **standlrt**. We can specify a set of values for the predictor variables for which we want to make predictions. We can do this for continuous variables as a set of **Values**, as a **Range,** as **Percentiles**, or **Nested Means** Here we use a range, postponing discussion of the other options until later.

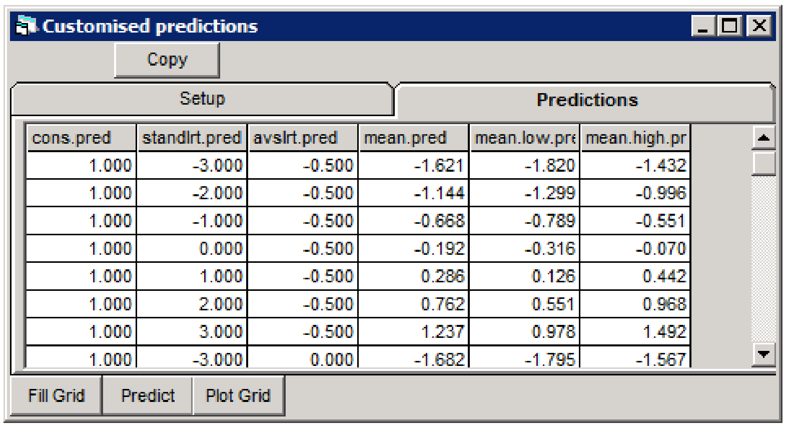
Note that, whether we specify values as a set of values, as a range, as percentiles, or as nested means, we are setting values for the original predictor variable. If for example we have centred the variable using one of the options in the **Specify Term** window, or via the CENT and ADDT commands, then we specify values for the uncentred variable, not for the centred variable. The same applies for other transformations such as polynomials, as long as the transformation was carried out as part of entering the variable into the model, via the **Specify Term** window or the ADDT command. MLwiN will display the name of the original, untransformed variable in the **Customised predictions** and will apply the necessary transformation to produce transformed values to use in calculating the prediction. This behaviour makes it easier for the user to specify a range of values and also enables plotting predictions against the original rather than the transformed variable.

2.1.1 Setting values as a range

The output column is set by default to the next free column (c13 in this case)

- Click the **Range** tab and specify **Upper Bound**: 3, **Lower**: −3, **Increment**: 1
- Click the **Done** button to close the **Values to include** window
- Using the same procedure, set the range for **avslrt** to be Upper Bound: 0.5, Lower: −0.5, Increment: 0.5
- Click **Fill Grid** and **Predict**
- Select the **Predictions** tab

The **Fill Grid** button creates a mini dataset with all combinations of the explanatory variable values you have specified. The **Predict** button takes the estimates for the parameters of the current model and applies them to the mini dataset to create predicted values with confidence intervals.

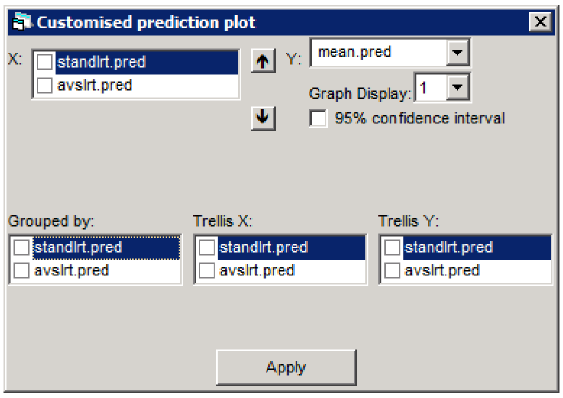


cons.pred	standlrt.pred	avslrt.pred	mean.pred	mean.low.pre	mean.high.pr
1.000	-3.000	-0.500	-1.621	-1.820	-1.432
1.000	-2.000	-0.500	-1.144	-1.299	-0.996
1.000	-1.000	-0.500	-0.668	-0.789	-0.551
1.000	0.000	-0.500	-0.192	-0.316	-0.070
1.000	1.000	-0.500	0.286	0.126	0.442
1.000	2.000	-0.500	0.762	0.551	0.968
1.000	3.000	-0.500	1.237	0.978	1.492
1.000	-3.000	0.000	-1.682	-1.795	-1.567

The columns in this prediction grid have all been created as columns in the worksheet. The columns at the left (**cons.pred**, **standlrt.pred**, **avslrt.pred**) contain the values of the explanatory variables that we specified and requested predictions for; **mean.pred** contains the predicted value of the response for each combination of explanatory variable values, and **mean.low.pred** and **mean.high.pred** contain the lower and upper bounds respectively of the confidence intervals for the predictions.

We can plot these results :

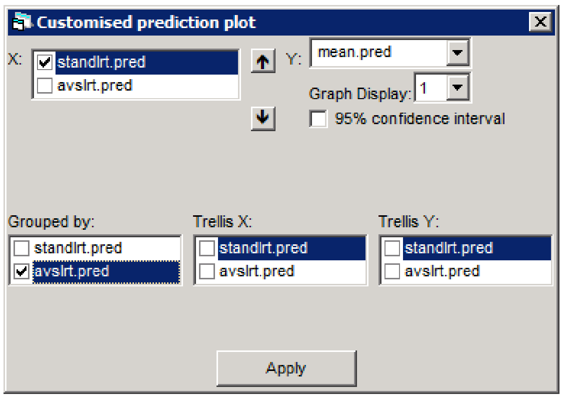
- Click **Plot Grid**



This screen has been designed to help construct plots from a multidimensional grid. Here we want

X: standlrt.pred
Y: mean.pred
Grouped by avslrt.pred

that is,



Customised prediction plot

X: ☑ standlrt.pred ☐ avslrt.pred

Y: mean.pred

Graph Display: 1

☐ 95% confidence interval

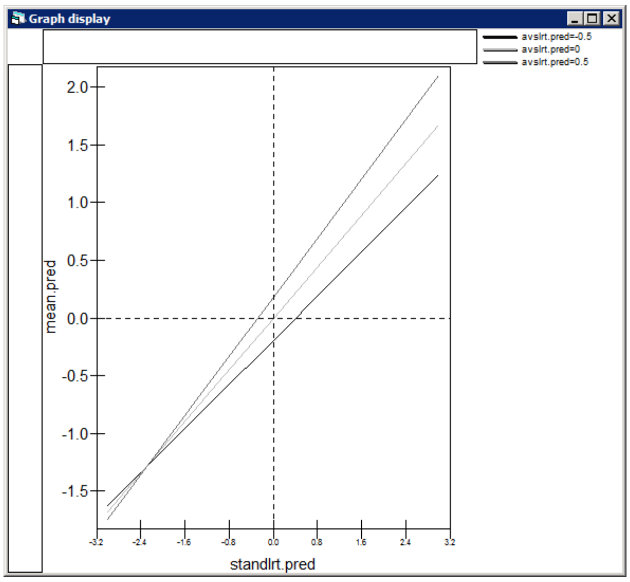
Grouped by: ☐ standlrt.pred ☑ avslrt.pred

Trellis X: ☐ standlrt.pred ☐ avslrt.pred

Trellis Y: ☐ standlrt.pred ☐ avslrt.pred

Apply

- Click **Apply**

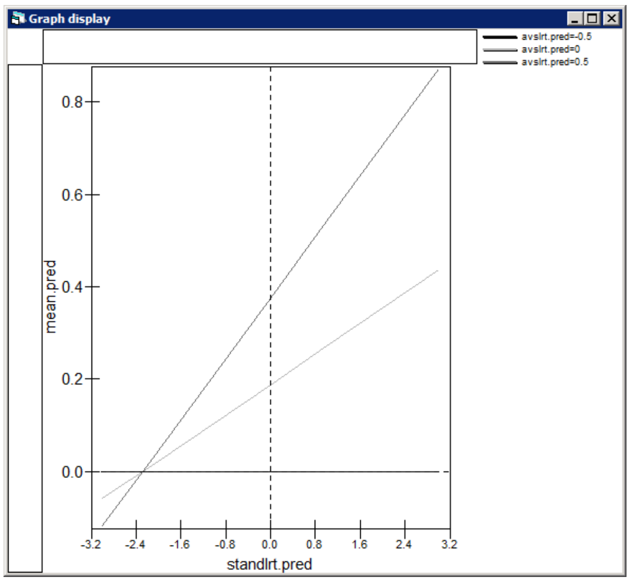


The settings for this plot are held in Display 1 of Customised graphs.

We can look at the differences (and the confidence intervals of the differences) from any reference group. For example if we wanted to look at the above plot as differences from **avslrt.pred** $= -0.5$, then we would

- Select the **Customised predictions** window
- Tick the **Differences** checkbox (click the **Setup** tab first if necessary)
- In the **from variable** drop-down list select **avslrt**
- In the box labelled **Reference value** type -0.5
- Click **Predict**

The graph will update automatically to look like this:



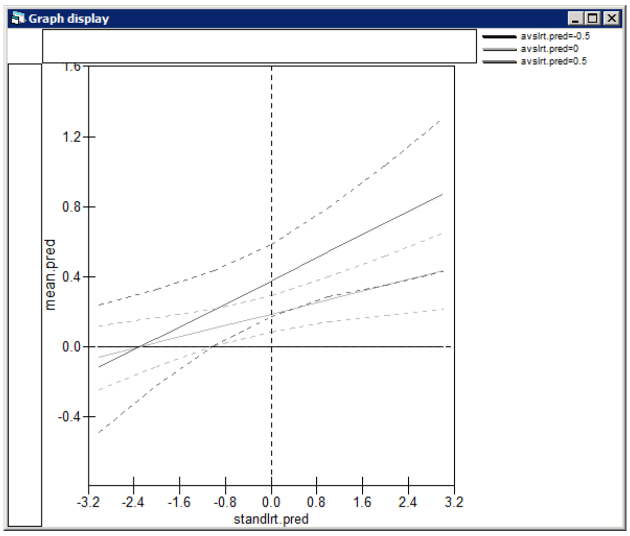
If we want to put confidence intervals around these differences then, in the **Customised prediction plot** window:

- Tick the **95% confidence interval** check box

Options appear allowing the confidence intervals to be drawn as bars or lines, with lines as the default.

- Click **Apply**
- Click the **OK** button on the warning message which may appear

to produce:



We now discuss the other possibilities for specifying the values of the explanatory variables for which you want predictions.

2.1.2 Specification using the Values tab

- Return to the **Customised Predictions** window
- Highlight **standlrt** and click the **Change Range** button
- Click the **Values** tab
- In the top pane type **−2** and click the **Add** button
- Add the values **0** and **3** in the same way
- Click **Done**

Clicking **Fill Grid** and **Predict** in the **Customised Predictions** window will now produce predictions as before but this time taking $\{-2, 0, 3\}$ as the set of values for **standlrt** instead of $\{-3, -2, -1, 0, 1, 2, 3\}$. Notice that using the **Values** tab automatically cleared the values previously specified using the **Range** tab.

Note also that the software will automatically check whether the values you specify are within the range of the variable and an error message will appear if you try to specify a value outside of this range.

2.1.3 Specification using the Percentiles tab

- Highlight **standlrt** and click the **Change Range** button
- Click the **Percentiles** tab
- In the top pane type **10** and click the **Add** button
- Add the percentiles **50** and **90** in the same way
- Click **Done**

The set of values for **standlrt** that will be used if **Fill Grid** and **Predict** are pressed is now (to 2 d.p.) $\{-1.28, 0.04, 1.28\}$. These are the 10th, the 50th and the 90th percentile of **standlrt** respectively. Again, notice that the previously specified values have been automatically removed.

Percentiles are a useful way of specifying values when we would like, for example, a prediction for those with a ‘low’ value of **standlrt**, those with a ‘mid’ value of **standlrt**, and those with a ‘high’ value of **standlrt** because they provide a convenient way of deciding what counts as ‘low’, ‘mid’ and ‘high’, although there is still some subjectivity involved: in this case we could equally well have decided to use, for example, the 25th, 50th and 75th percentiles, which would give less extreme values for ‘low’ and ‘high’.

We need not necessarily use three percentiles; we can specify as many as we like, though generally a small number of percentiles will be quite sufficient for what we want here.

2.1.4 Specification using the Nested Means tab

Nested means are another way of dividing a numerical variable into groups; again they provide a convenient means of choosing, for example, a 'low', a 'mid' and a 'high' value (if you use 4 groups). Simply put, a set of nested means which divides the values into 4 groups is doing the same thing as quartiles (i.e. the 25th, 50th and 75th percentiles) but using the mean instead of the median. For quartiles, the 2nd quartile (or 50th percentile) is the median of the variable. The 1st quartile (or 25th percentile) is the median of the values between the minimum and the 2nd quartile, and the 3rd quartile (or 75th percentile) is the median of the values between the 2nd quartile and the maximum. In the same way, to get four groups using nested means we first calculate the mean of the variable and call this MidMean, then calculate the mean of the values between the minimum and MidMean and call this LowMean, and calculate the mean of the values between MidMean and the maximum and call this HighMean; LowMean, MidMean and HighMean are then the divisions between our four groups.

More generally, a frequency distribution is balanced about its mean, and this forms an obvious point of division to give two groups; each of these classes may be subdivided at its own mean; and so on, giving 2, 4, 8, 16, . . . , classes.

Evans (1977)(1) claims this approach has desirable 'adaptive' properties

"Since means minimize second moments (sums of squared deviations), they are the balancing points of the part of the scale which they subdivide, with respect to both magnitude and frequency. Class intervals thus defined are narrow in the modal parts of a frequency distribution and broad in the tails. Extreme values are not allowed to dominate, but they do influence the positions of means of various orders so that the less closely spaced the values in a given magnitude range, the broader the classes. For a rectangular frequency distribution, nested means approximate the equal-interval or percentile solutions; for a normal one, they approximate a standard deviation basis; and for a J-shaped, a geometric progression. Hence nested means provide the most robust, generally applicable, replicable yet inflexible class interval system."

To specify values using the **Nested Means** tab,

- Highlight **standlrt** and click the **Change Range** button
- Click the **Nested Means** tab
- To get 4 groups, specify **1 level of nesting**

- Click **Done**

The three values that will be used for **standlrt** in predictions are now the cut points of the four groups: the overall mean, the mean of values between the minimum and the overall mean, and the mean of values between the overall mean and the maximum.

We could also have obtained 7 values (the cut points of 8 groups) by specifying 2 levels of nesting, or 15 values (the cut points of 16 groups) by specifying 3 levels of nesting, and so on.

2.1.5 Specifying values for categorical variables

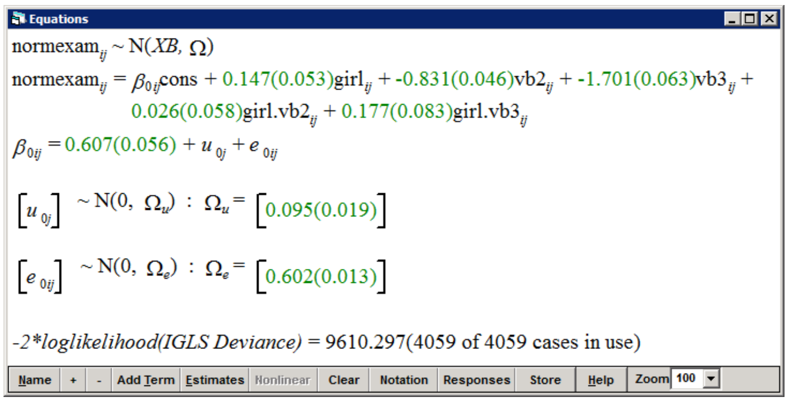
Let's set up a model with categorical predictors. This can be done in the Equations window or via commands in the Command Interface window. First let us recode the variable **girl**:

- In the **Names** window, select **girl** and click the **Edit name** button
- Change the name of **girl** to 'gender'
- Click the **Toggle Categorical** button
- Click the **View** button in the categories section
- Change the names of the categories from **gender_0** to 'boy' and from **gender_1** to 'girl'
- Click **OK** to close the **Set category names** window

We will use as our predictors **gender** and **vrband,** which is a categorised prior ability measure on pupils (vb1 = high, vb2 = mid, vb3 = low). Type the commands in the left hand column of the table below in the **Command interface** window

Command	**Explanation**
`CLEAR`	clear the current model
`IDEN 1 'student' 2 'school'`	declare multilevel structure
`RESP 'normexam'`	declare response variable
`EXPL 1 'cons' 'gender' 'vrband'`	specify explanatory variables
`ADDT 'gender' 'vrband'`	add interaction term
`SETV 2 'cons'`	specify random intercept at level 2
`SETV 1 'cons'`	specify constant variance at level 1
`STAR 1`	estimate model updating GUI

Note that the command STAR 1 has the same effect as pressing the **Start** button at the top left of the screen. This will produce the following results in the **Equations** window



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

$\text{normexam}_{ij} = \beta_{0ij}\text{cons} + 0.147(0.053)\text{girl}_{ij} + -0.831(0.046)\text{vb2}_{ij} + -1.701(0.063)\text{vb3}_{ij} + 0.026(0.058)\text{girl.vb2}_{ij} + 0.177(0.083)\text{girl.vb3}_{ij}$

$\beta_{0ij} = 0.607(0.056) + u_{0j} + e_{0ij}$

$\left[u_{0j}\right] \sim \text{N}(0,\ \Omega_u)\ :\ \Omega_u = \left[0.095(0.019)\right]$

$\left[e_{0ij}\right] \sim \text{N}(0,\ \Omega_e)\ :\ \Omega_e = \left[0.602(0.013)\right]$

*-2*loglikelihood(IGLS Deviance)* = 9610.297(4059 of 4059 cases in use)

- Open the **Customised predictions** window

A message appears at the top of the window:
"Prediction is out of date, specification disabled until it is cleared"

This is because the current prediction grid refers to a previous model. Indeed looking at the list of explanatory variables in the **Customised predictions** window, you will see the variables **standlrt** and **avslrt** listed. When the main effects in a model are altered the prediction grid is flagged as being out of date. To proceed with a prediction for the new model we must clear the current prediction grid specification

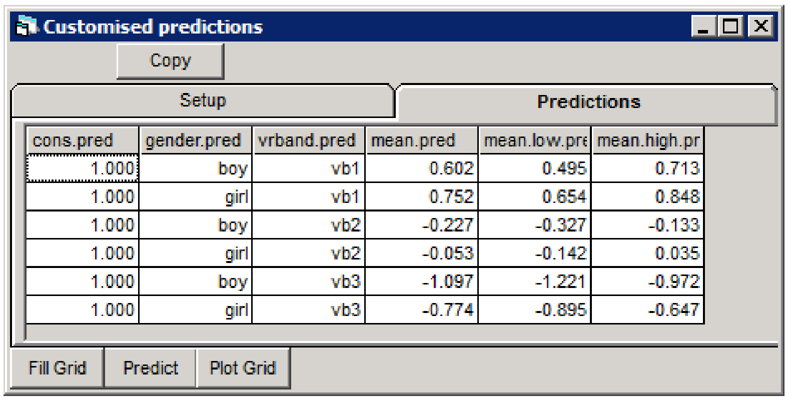
- Press the **Clear** button in the **Customised predictions** window

This will clear any columns or graphs referred to by the prediction grid and set the prediction grid to refer to the current model. You will now see that the list of explanatory variables in the **Customised predictions** window is up to date. In the **Customised predictions** window:

- Highlight **gender**
- Click **Change Range**
- Select **Category**
- Select **boy** and **girl** in the **Values to include** list
- Click **Done**
- Highlight **vrband** in the **Customised predictions** window
- Click **Change Range**
- Click **Category**

- Select **vb1**, **vb2** and **vb3**
- Click **Done**
- Click the **Fill Grid** and **Predict** buttons in the **Customised predictions** window
- Select the **Predictions** tab in the **Customised predictions** window

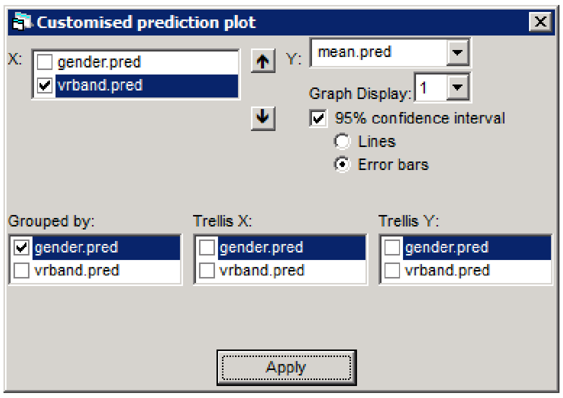
This shows us the predicted values and confidence intervals for the 6 combinations of **gender** and **vrband**



cons.pred	gender.pred	vrband.pred	mean.pred	mean.low.pre	mean.high.pr
1.000	boy	vb1	0.602	0.495	0.713
1.000	girl	vb1	0.752	0.654	0.848
1.000	boy	vb2	-0.227	-0.327	-0.133
1.000	girl	vb2	-0.053	-0.142	0.035
1.000	boy	vb3	-1.097	-1.221	-0.972
1.000	girl	vb3	-0.774	-0.895	-0.647

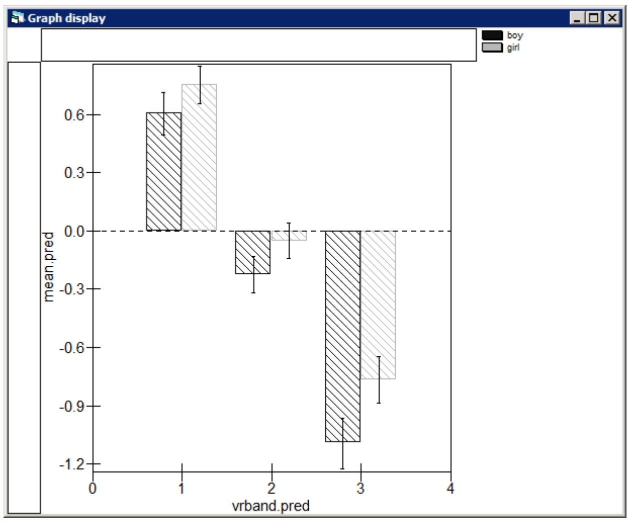
Note that, because simulation is used to calculate these values, these numbers may be slightly different every time.

Plotting this out using the **Customised prediction plot** window filled in as follows:



X: [ ] gender.pred, [✓] vrband.pred
Y: mean.pred
Graph Display: 1
[✓] 95% confidence interval
( ) Lines
(●) Error bars
Grouped by: [✓] gender.pred, [ ] vrband.pred
Trellis X: [ ] gender.pred, [ ] vrband.pred
Trellis Y: [ ] gender.pred, [ ] vrband.pred

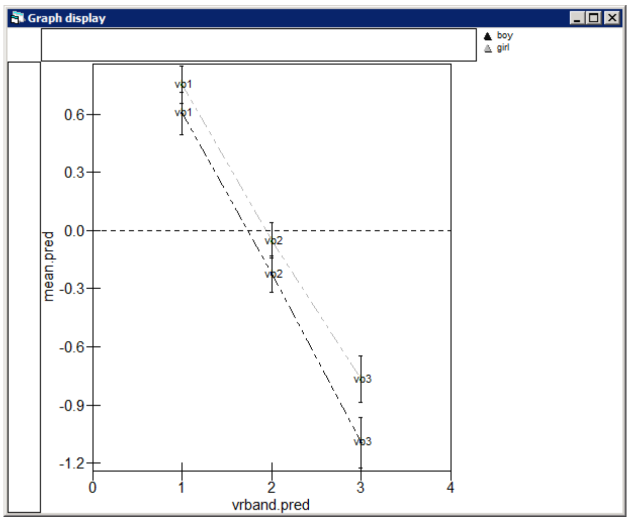
gives this:



This shows girls doing uniformly better than boys across all 3 levels of **vrband**. Recall that **vrband** is a categorised prior ability measure on pupils (vb1 = high, vb2 = mid, vb3 = low). The default choice for displaying data against a categorical x variable is to plot the data as bars. We can change this or other characteristics of the plots produced by the **Customised prediction plot** window by editing the **Customised graph** window for the appropriate graph display. For example, suppose we wanted to draw the above relationship using line + point instead of bars. Then

- Select **Customised Graph(s)** from the **Graphs** menu
- Select **line + point** from the drop-down list labelled **plot type**
- Click **Apply**

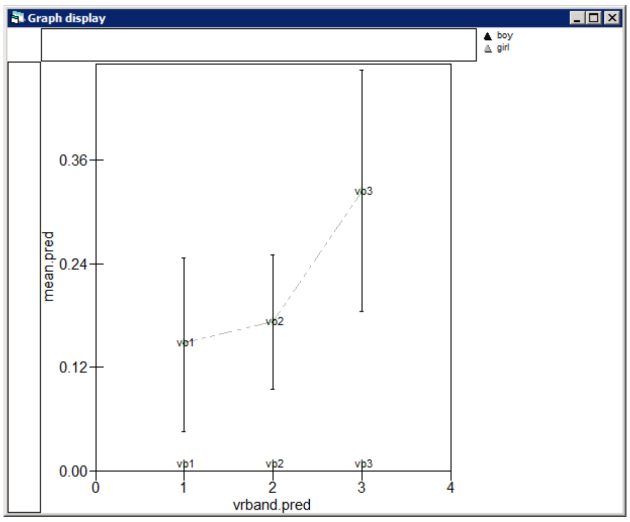
This produces:



Both these graphs show girls outperforming boys at all three levels of **vrband**. Looking at the graphs (and noting in which cases the error bars overlap) it might appear that this gender difference is significant for **vb3** children, but not for children in **vb2** and **vb1.** However, care is needed here. The graphs are showing 95% confidence intervals around the predicted *means* for each of the 6 groups. If we are interested in making inferences about how the gender *difference* changes as a function of **vrband**, we must request confidence intervals for these differences:

- Select the **Customised predictions** window
- Tick the **Differences** checkbox
- In the **from variable** drop-down list select **gender**
- In the **Reference value** drop-down list select **boy**
- Click **Predict**

The graph will update



which shows that the 95% confidence intervals for the gender differences do not overlap zero for **vb1, vb2** or **vb3**, and thus there is a significant gender difference for each of the three categories of **vrband**.

2.1.6 Commands for building customised predictions for Normal response models

`PGDE` : clears the current prediction grid — it should be used before constructing a new prediction grid

PGRI constructs a prediction grid:

PGRI {Continuous **C** contmode = **N**, contmode values **N**..**N**} {categorical **C** catmode **N**, catmode **N**..**N**} Outputcols **C**..**C**

contmode **N**
= 0 list of values: **N**..**N**
= 1 range: upper bound **N**, lower **N**, increment **N**
= 2 centiles: **N**..**N**
= 3 nested means: level of nesting **N**

catmode **N**
= 1 category numbers **N** ... **N**
= 2 all categories

Example:

```
▶ PGRId 'cons' 0 1 'standlrt' 1 3 -3 1 'schgend' 2 C22 C23
  C24
```

PREG calculates predictions for the prediction grid currently set up.

For Normal response models:

`PREG` confidence interval **N** mean **C** lower **C** upper **C**, differences **N** (differences for column **C**, differences from category **N**), multivariate **N** (respcol **C**), coverages {level **N** coverage range **N** low **C** up **C**} ...

Examples:
(1)

```
▶ PREG 95 mean c100 lower c101 upper c102, difference 0,
  multivariate 0
```

Given PGRID established as above, predicted mean +/- 95 percentile output to c100 c101 c102, no differences selected

(2)

```
▶ PREG 95 mean c100 lower c101 upper c102, difference 0,
  multivariate 1 'normexam'
```

Given PGRID established for a multivariate response model, performs same task as example (1) above; predictions are for the response 'normexam'.

(Note that we describe later how to make predictions for multivariate models using the customised predictions window).

(3)

```
► PREG 95 mean c100 lower c101 upper c102, difference 1
  'schgend' 1, multivariate 1 'normexam'
```

As (2) above but predictions are differenced from **schgend** category 1

(4)

```
► PREG 95 mean c100 lower c101 upper c102, difference 1
  'schgend' 1, multivariate 1 'normexam' 2 99 c103 c104
```

As (3) above but also calculate 99% coverage for predictions based on level 2 variance; lower and upper coverage values to c103 and c104

2.1.7 Commands for plotting customised predictions

The PLTPrediction command is used for displaying predictions and performs the same task as the **Customised prediction plot** window; though it can also be used more generally to plot data other than predictions since it just plots one column against another, with confidence intervals if supplied, with the options to plot the data by groups and to split the plot into two separate graphs (horizontally or vertically), with data plotted on one graph or the other according to values of a supplied variable:

`PLTPrediction` in graph display **N**, dataset (X,Y),
0 (do nothing) or grouped by values in **C**,
0 (do nothing) or split into separate graphs across rows of a trellis according to values in **C**,
0 (do nothing) or split into separate graphs across columns of a trellis according to values in **C**,
{confidence intervals: lower in **C**, upper in **C**, plot style 10=bar, 11=line}

Note that if you do not already have the graph display window open then you will need to go to the **Customised graph** window and click **Apply** after entering the command.

Examples:

(1)

```
▶ PLPT 2 c1 c2 0 0 0 c3 c4 11
```

Sets up graph display 2: $x = c1$ $y = c2$, lower and upper confidence intervals are in c3 and c4, plot confidence intervals as lines

(2)

```
▶ PLPT 2 c1 c2 ‘social_class' 0 0
```

Sets up graph display 2: $x = c1$ $y = c2$, one line for each social class, don’t display confidence intervals

(3)

```
▶ PLPT 2 c1 c2 'social_class.pred' 'gender.pred' 0 c3 c4 11
```

Sets up graph display 2: x = c1 y = c2, one line for each social class, repeat plot in a 2 x 1 graph trellis for **gender** = 0 in first pane and **gender** = 1 in second pane (i.e. split into two plots arranged vertically, with the upper plot including data points for which **gender** = 0 and the lower plot including data points for which **gender** = 1), lower and upper confidence intervals are in c3 and c4, plot confidence intervals as lines

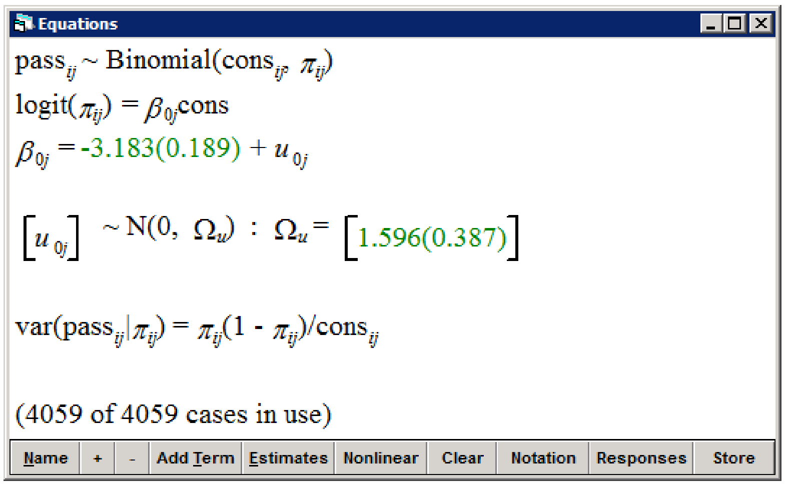
2.2 Binomial models

2.2.1 Predicting mean and median

To give ourselves a binary response variable so we can demonstrate prediction for binomial models, let’s create a dichotomised variable from the Normalised response in the **tutorial** dataset. Responses with a value of greater than or equal to 1.5 we will set to 1 otherwise we will set the dichotomised variable to 0. To do this, type in the **Command interface** window

```
▶ calc c11 = 'normexam' >= 1.5
▶ name c11 'pass'
```

Now set up the following multilevel binomial model and estimate the model with the non-linear options set to 2nd order PQL estimation :


Equations

$\text{pass}_{ij} \sim \text{Binomial}(\text{cons}_{ij}, \pi_{ij})$

$\text{logit}(\pi_{ij}) = \beta_{0j}\text{cons}$

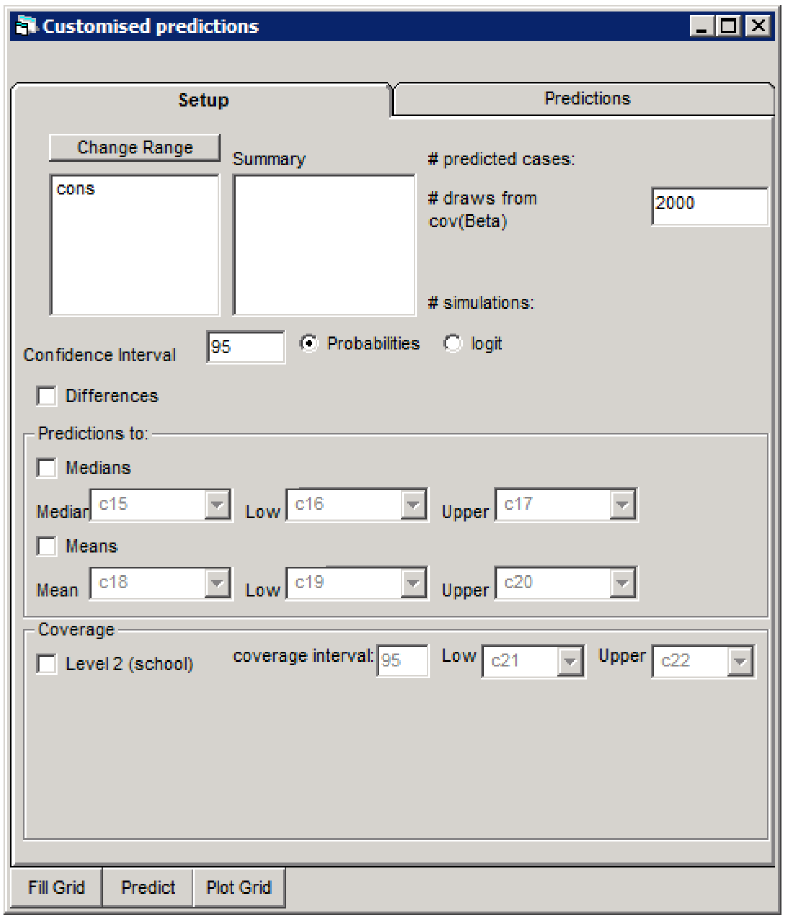
$\beta_{0j} = -3.183(0.189) + u_{0j}$

$\left[u_{0j}\right] \sim \text{N}(0, \Omega_u) : \Omega_u = \left[1.596(0.387)\right]$

$\text{var}(\text{pass}_{ij}|\pi_{ij}) = \pi_{ij}(1 - \pi_{ij})/\text{cons}_{ij}$

(4059 of 4059 cases in use)

The Customised predictions window now looks like this:



# predicted cases:	
# draws from cov(Beta)	2000
# simulations:	
Confidence Interval	95
[✓ Probabilities] logit	
Differences	☐

Predictions to:

☐ Medians					
Median	c15	Low	c16	Upper	c17
☐ Means					
Mean	c18	Low	c19	Upper	c20

Coverage

☐ Level 2 (school)	coverage interval:	95	Low	c21	Upper	c22

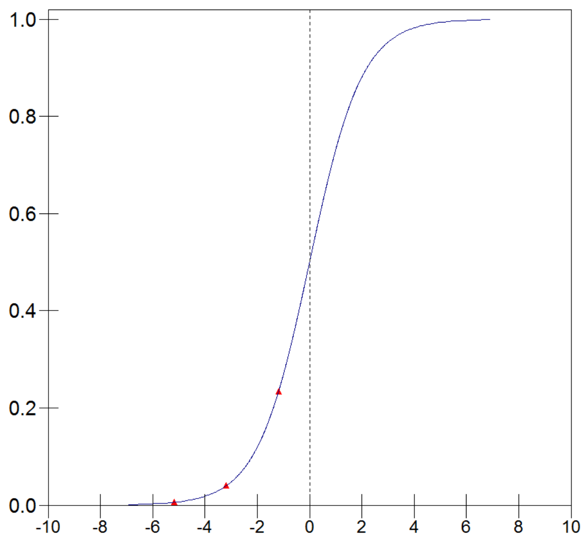
The **Customised predictions** window for binomial response models looks similar to that for Normal response models. However there are a few changes. You can choose whether predictions should be made on the logit or probability scales and also you can ask for these predictions to be for the mean or median value of the prediction distribution.

What does prediction distribution mean? This data is based on 65 schools. The level 2 variance of 1.596 on the logit scale is the between school variance, in the population from which our schools are sampled, of the school means (on the logit scale). Suppose we now ask the question of what the mean and median pass rates are on the probabilty scale, for pupils attending three schools (of identical size) with values of $u_{0j} = \{-2, 0, 2\}$. This is a school from either end and the middle of the distribution of schools. We obtain p_k, the pass rate on the probability scale, for each school k by taking the antilogit of the model equation $-3.18 + u_{0k}$ (putting in the appropriate value for u_{0k})

$$\begin{aligned} p_k &= \{\text{antilogit}(-3.18 - 2), \text{antilogit}(-3.18), \text{antilogit}(-3.18 + 2)\} \\ &= \{0.006, 0.040, 0.234\} \\ \text{mean}(p_k) &= 0.093 \\ \text{median}(p_k) &= 0.040 \end{aligned}$$

The thing to note is that if we take the antilogit of the mean of our three logit values, that is antilogit$(-3.18) = 0.040$, this is equal to the median of our three probabilities but *not* the mean of the three probabilities (0.0918). This is because in this case before we take antilogits the mean of our three values equals the median, and the median of the antilogits of a set of values is always equal to the antilogit of the median of the set of values. The mean of the antilogits of a set of values on the other hand will in general *not* be equal to the antilogit of the mean of the set of values.

This comes about because of the non-linearity of the logit transformation as we can see from the graph below, which graphs logit(p) and places our three schools as the points marked by triangles on the line.



When we fit a multilevel model and take the antilogit of a fixed part predictor (i.e. a sum of fixed part coefficients multiplied by particular values of their associated explanatory variables), this gives us the median (not the mean) probability for the prediction case consisting of those values of the explanatory variables. Again, this is because before we take antilogits, median =

mean (since we are dealing with a normal distribution), and antilogit(median) = median(antilogits). From the above multilevel model we get a prediction of -3.18 on the logit scale which corresponds to a median probability of passing of 0.040. We may however want to know what the mean probability is of passing, not taking account of only three schools, but allowing for the whole distribution of schools. The mean cannot be directly calculated from the model in the same way that the median can; but we can estimate it using simulation. We know that the distribution of all schools on the logit scale is $\mathrm{N}(-3.183, 1.596^{0.5})$. Thus the following will give us an estimate of the mean pass rate

1. Simulate the value on the logit scale for a large number of schools, say 1000, from the distribution $\mathrm{N}(-3.183, 1.596^{0.5})$. That is:

$$Z_j = \mathrm{N}(-3.183, 1.596^{0.5}), \quad j = 1, \ldots, 1000$$

2. Calculate the predicted probability for each of our 1000 schools and calculate their mean. That is, $\overline{\text{antilogit}(Z_j)}$.

We can do this in MLwiN using the following commands (click the **Output** button on the Command interface window before typing them in order to properly see the output at the end)

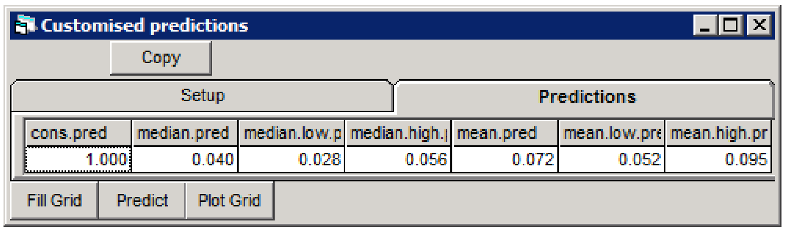
Command	**Explanation**
`seed 8`	set random number seed (so we can replicate our results)
`nran 1000 c100`	pick 1000 values from N(0,1)
`calc c100=c100*1.596^0.5-3.183`	transform to $\mathrm{N}(-3.183, 1.596^{0.5})$
`calc c101=alog(c100)`	generate 1000 probabilities
`aver c101`	calculate their mean

This produces an estimate of the mean pass rate as 0.072. In this example our prediction case is very simple, it is simply the unconditional mean pass rate (i.e. the mean pass rate taking no explanatory variables into account); if we calculate that directly (by taking the mean of the variable **pass**) we get a value of 0.070, so we can see our estimate is very good. In more complex conditional predictions, for example the mean pass rate for girls with intake scores of 1 in girls' schools, we may have very few (or in fact no) individuals exactly fitting our prediction case and a reasonable empirical estimate of the mean for that prediction group is not available. However, the model based estimate is available.

All the predictions and confidence intervals calculated with the out of sample predictions window are derived by simulation, in a similar fashion to the example we just typed in the commands for. The detailed simulation algorithms for a range of model types are given in Appendix 1 of this document.

These median and mean predictions are often referred to as cluster specific and population average predictions respectively. Instead of typing in the above commands, we can obtain these simply using the **Customised predictions** window:

- Select **Probabilities**
- Check boxes to request predicted **Medians** and **Means**
- Click **Fill Grid**
- Click **Predict**
- Select the **Predictions** tab



cons.pred	median.pred	median.low.p	median.high.|	mean.pred	mean.low.pre	mean.high.pr
1.000	0.040	0.028	0.056	0.072	0.052	0.095

We see that the median ('median.pred') and mean ('mean.pred') predictions are as expected. Note that these results are from simulation algorithms where we have drawn a particular number of simulated values.

If you click on the **Setup** panel in the **Customised predictions** window you will see

# predicted cases: 1
# draws from cov(Beta) 2000
# nested draws from cov(u) 1000
# simulations 2000000

The number of predicted cases is 1 because we are predicting for the unconditional mean only. When we typed in commands to estimate the mean probability and generated 1000 draws from $\mathrm{N}(-3.183, 1.596^{0.5})$, we always used the same value for $\hat{\beta}_0$ of -3.183. In fact, s.e.$(\hat{\beta}_0)$ = 0.189 and the full simulation procedure as implemented by the Customised predictions window is as follows

1. Simulate K= 2000 values of
   $\beta_{0k} \sim \mathrm{N}(-3.183, 0.189^{0.5}), \quad k = 1 : K$
2. For each value of β_{0k} simulate j = 1000 values of
   $\beta_{0kj} \sim \mathrm{N}(\beta_{0k}, 0.189^{0.5}), \quad j = 1 : J$

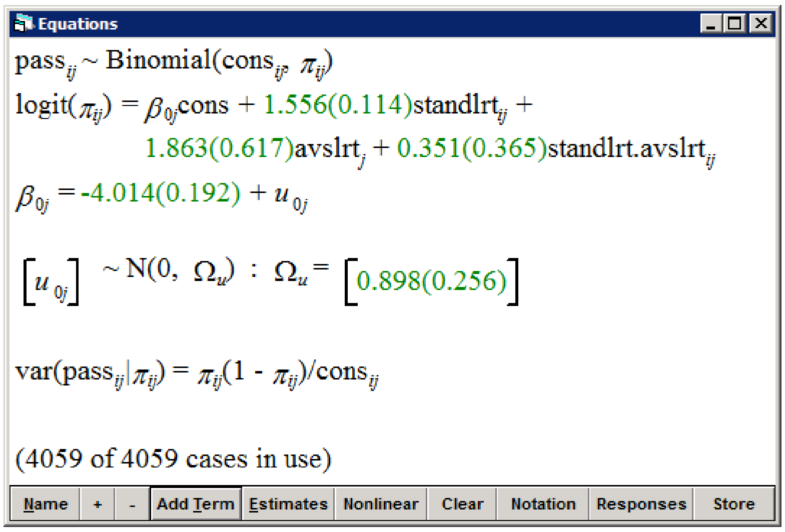
3. Then let $p_{kj} = \text{antilogit}(\beta_{0kj})$ and for each value of k calculate $p_k = \frac{1}{J}\sum_{j=1}^{J} p_{kj}$

4. Finally we calculate the mean probability as $p = \frac{1}{K}\sum_{k=1}^{K} p_k$

This actually involves $K \times J = 2000000$ simulation draws and the results in this case are identical to the simplified procedure where we typed in commands and assumed a fixed value for $\hat{\beta}_0$.

2.2.2 Predictions for more complex Binomial response models

Let's fit a model where the probability of passing is a function of pupil intake score (**standlrt**), peer group ability (**avslrt**) and an interaction of pupil intake score and peer group ability. Note the interaction effect between avslrt and standlrt is not significant; we leave it in the model for the purposes of demonstrating the graphing of interactions, particularly for log odds ratios in the next section.



$\text{pass}_{ij} \sim \text{Binomial}(\text{cons}_{ij}, \pi_{ij})$

$\text{logit}(\pi_{ij}) = \beta_{0j}\text{cons} + 1.556(0.114)\text{standlrt}_{ij} + 1.863(0.617)\text{avslrt}_j + 0.351(0.365)\text{standlrt.avslrt}_{ij}$

$\beta_{0j} = -4.014(0.192) + u_{0j}$

$\begin{bmatrix} u_{0j} \end{bmatrix} \sim \text{N}(0,\ \Omega_u) \ : \ \Omega_u = \begin{bmatrix} 0.898(0.256) \end{bmatrix}$

$\text{var}(\text{pass}_{ij}|\pi_{ij}) = \pi_{ij}(1 - \pi_{ij})/\text{cons}_{ij}$

(4059 of 4059 cases in use)

In the **Customised predictions** window, set

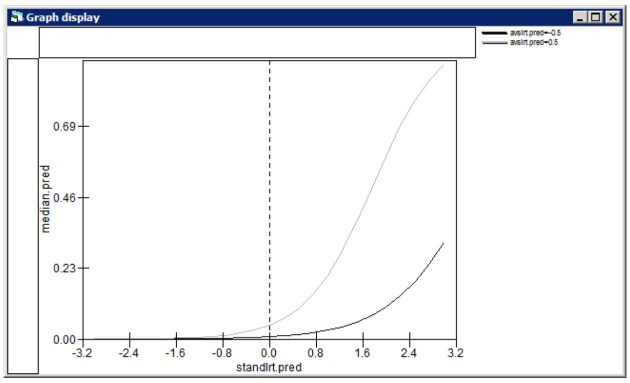
- **standlrt** = range $\{3, -3, 0.25\}$ (note we will use this notation from now on for convenience; it means 'define the values of **standlrt** using the **Range** tab; set Upper Bound = 3, Lower = -3, Increment = 0.25')
- **avslrt** = values $\{0.5, -0.5\}$
- Select **Median** predictions
- Select predictions for **Probabilities**

- Click **Fill grid**
- Click **Predict**
- Click **Plot grid**

In the **Customised prediction plot** window

- Select **Y:** median.pred
- Select **X:** standlrt.pred
- Select **Grouped by:** avslrt.pred
- Click **Apply**

Which produces:

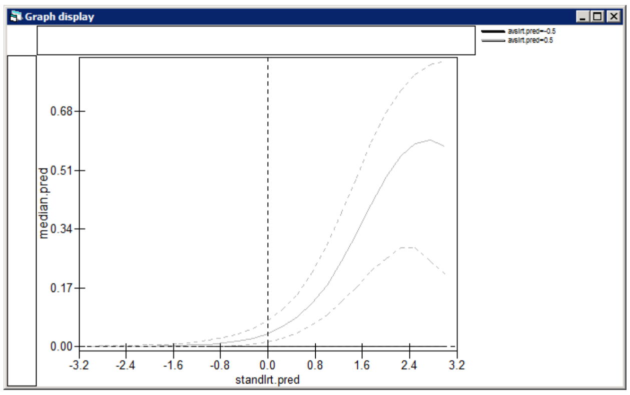


The graph shows how the probability of passing increases as pupil intake ability increases, with the increase being stronger for those in high ability peer groups. We now explore the differences between low and high ability peer groups as functions of **standlrt** and assess if and where those differences are significant. In the **Customised predictions** window

- Select **Differences**; **from variable**: avslrt **Reference value**: -0.5
- Press the **Predict** button

In the **Customised prediction plots** window

- Select **95% confidence interval**:
- Click **Apply**

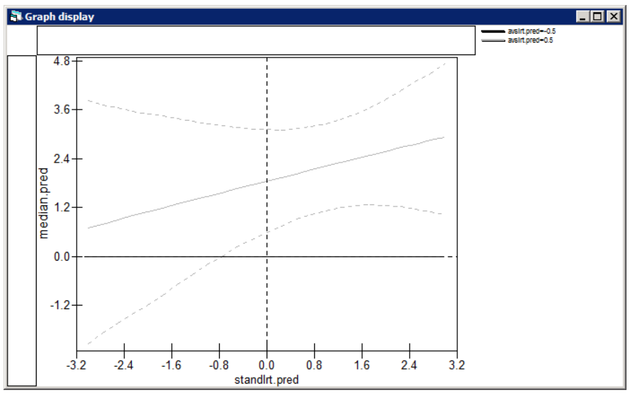


We can see that after **standlrt** scores become greater than around -0.5, the probability of passing for those in the high ability group becomes significantly different from the low ability group.

2.2.3 Log Odds ratios

Return to the **Customised predictions** window

- Select **logit**
- Click **Predict**



The variable **standlrt.pred** contains the requested values range $\{-3, 3, 0.25\}$. The graph plots

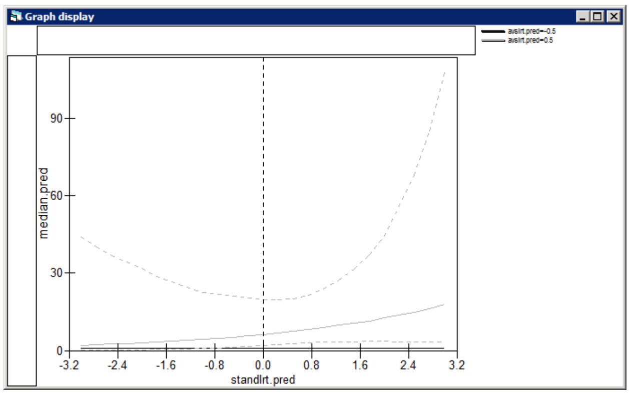
$$\log\left(\frac{\text{odds passing}|\textbf{avslrt.pred} = 0.5, \textbf{standlrt.pred}_i}{\text{odds passing}|\textbf{avslrt.pred} = -0.5, \textbf{standlrt.pred}_i}\right),$$

$$\textbf{standlrt.pred}_i = \{-3, -2.75, \ldots, 3\}$$

that is, the log odds ratio of passing for high versus low peer groups as a function of pupil intake score. We may want to view the graph as odds ratios, rather than log odds ratios. Looking at the **Names** window we see that the log odds ratio and its upper and lower bounds are in the columns named **median.pred**, **median.low.pred** and **median.high.pred**, that is c18-c20. To create a graph of odds ratios type the command

```
▶ expo 'median.pred' 'median.low.pred' 'median.high.pred'
  'median.pred' 'median.low.pred' 'median.high.pred'
```

A message will appear asking if you want to clear the prediction grid; click "No".



2.2.4 Customised prediction commands for discrete response models

`PREG` confidence interval **N**,
predict mean values **N** (mean values output set **C** lower **C** upper **C**),
predict median values **N** (median **C** lower **C** upper **C**)
prediction scale **N**,
differences **N** (diffcol **C**, diff reference **C**)
multivariate **N** (respcol **C**)
coverages {level **N** coverage range **N** low **C** up **C**} ..

This is an extension of the PREG command for Normal responses. The differences are

predict mean values **N**=
0: no mean value output set to follow
1: mean value output set required
(mean value output set: mean to **C**, lower confidence interval for mean to **C**, upper confidence interval for mean to **C**)

predict median values **N** =
0: no median value output set to follow
1: median value output set required
(median value output set: median to **C**, lower confidence intervals for median, upper confidence intervals for median)

Prediction scale **N** =
0: for raw response scale
2: for link function attached to model
Note raw response scale is probability (binomial, multinomial models) or counts (negative binomial, Poisson models)

Examples

(1)

```
▶ PREG 95 0 1 c100 c101 c102 0 0 0
```

If we have a binary response model set up, then the above command evaluates the current PGRID, calculating median (i.e. cluster specific) predictions on the probability scale, and writes the predicted medians and their lower and upper 95% confidence intervals to c100, c101, c102

(2)

```
▶ PREG 95 0 1 c100 c101 c102 1 'schgend' 1 0 0
```

As (1) above but predictions are differenced from **schgend** category 1

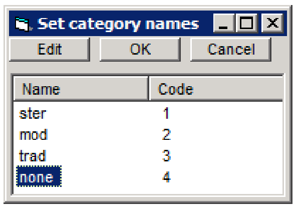
2.3 Multinomial models

2.3.1 Unordered Multinomial Models

We will take the contraception dataset used in Chapter 10 of the User's Guide:

- Open **bang.ws**
- Select the variable **use4** in the **Names** window
- Click the **View** button in the categories section

Set the 4 categories to be :

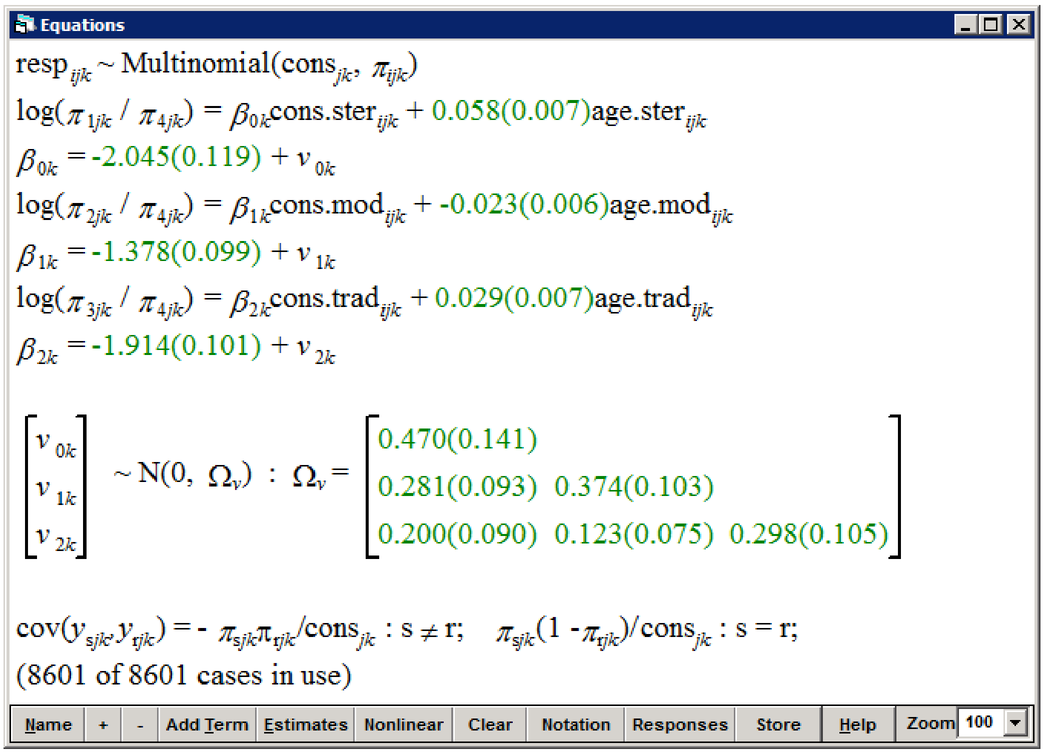


Name	Code
ster	1
mod	2
trad	3
none	4

Let's use the Command interface window to set up a multinomial model, where the log odds of using different types of contraception compared with no contraception are allowed to vary as a function of age.

Command	**Explanation**
`MNOM 0 'use4' c13 c14 4`	the reference category is 4- none
`NAME c13 'resp' c14 'resp_cat'`	
`IDEN 1 'resp_cat' 2 'woman' 3 'district'`	specify level IDs
`ADDT 'cons'`	add 3 intercepts — one for each log odds ratio
`ADDT 'age'`	add 3 age coefficients — one for each log odds ratio
`SETV 3 'cons.ster' 'cons.mod' 'cons.trad'`	specify random intercepts across districts
`DOFFs 1 'cons'`	specify denominator
`LINEarise 1 2`	select PQL order 2

Running this model gives:



$$\text{resp}_{ijk} \sim \text{Multinomial}(\text{cons}_{jk}, \pi_{ijk})$$

$$\log(\pi_{1jk} / \pi_{4jk}) = \beta_{0k}\text{cons.ster}_{ijk} + 0.058(0.007)\text{age.ster}_{ijk}$$

$$\beta_{0k} = -2.045(0.119) + v_{0k}$$

$$\log(\pi_{2jk} / \pi_{4jk}) = \beta_{1k}\text{cons.mod}_{ijk} + -0.023(0.006)\text{age.mod}_{ijk}$$

$$\beta_{1k} = -1.378(0.099) + v_{1k}$$

$$\log(\pi_{3jk} / \pi_{4jk}) = \beta_{2k}\text{cons.trad}_{ijk} + 0.029(0.007)\text{age.trad}_{ijk}$$

$$\beta_{2k} = -1.914(0.101) + v_{2k}$$

$$\begin{bmatrix} v_{0k} \\ v_{1k} \\ v_{2k} \end{bmatrix} \sim \text{N}(0, \Omega_v) \; : \; \Omega_v = \begin{bmatrix} 0.470(0.141) & & \\ 0.281(0.093) & 0.374(0.103) & \\ 0.200(0.090) & 0.123(0.075) & 0.298(0.105) \end{bmatrix}$$

$$\text{cov}(y_{sjk}, y_{rjk}) = -\pi_{sjk}\pi_{rjk}/\text{cons}_{jk} : s \neq r; \quad \pi_{sjk}(1 - \pi_{rjk})/\text{cons}_{jk} : s = r;$$

(8601 of 8601 cases in use)

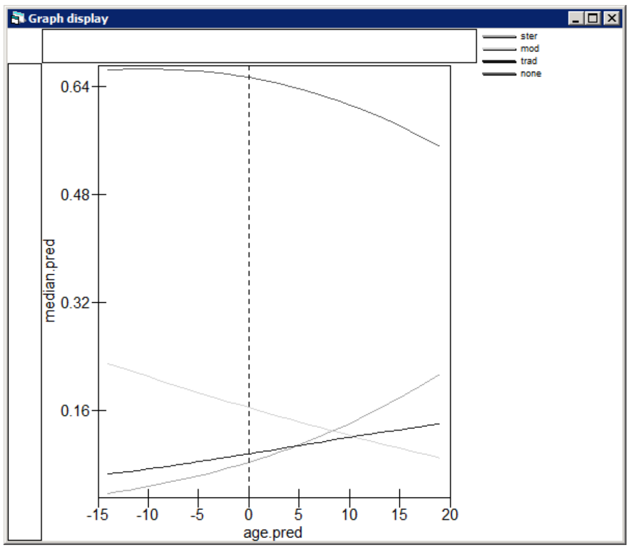
In the **Customised predictions** window :

- Set **age** to **range** {19, −14, 1}
- Select **Probabilities**
- Select **Medians**
- Click the **Fill grid, Predict** and **Plot Grid** buttons

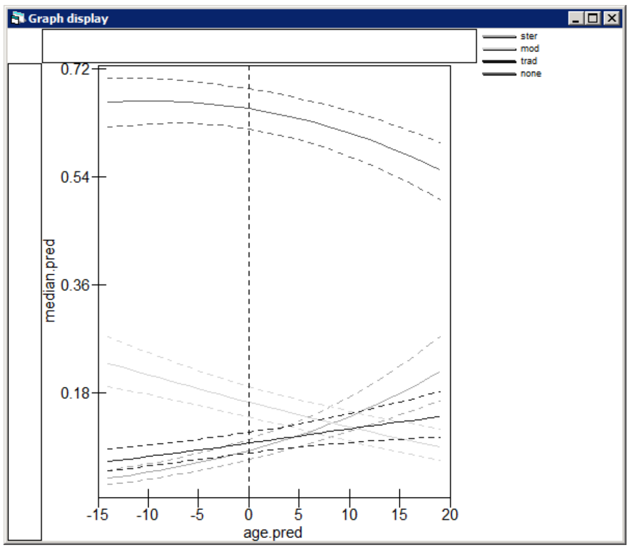
In the **Customised prediction plot** window

- Select **Y:** median.pred, **X**: age.pred, **Grouped by:** use4.pred
- Click Apply

which produces



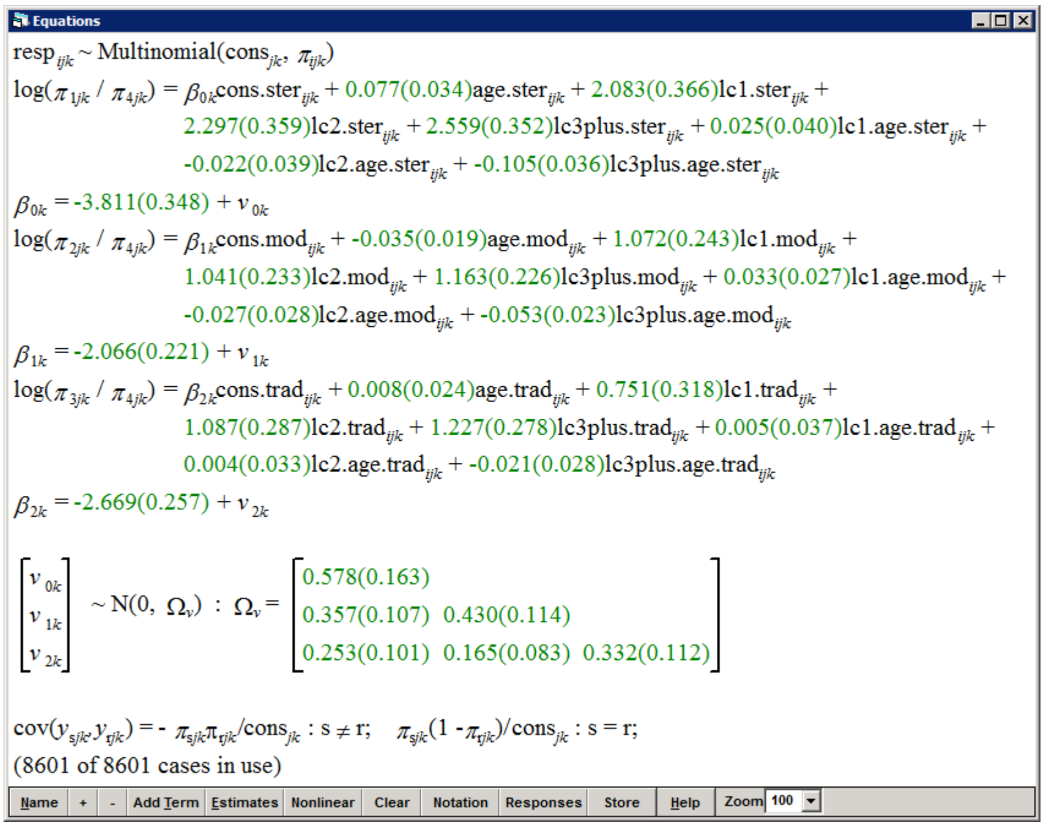
or with error lines



Let's elaborate the model adding number of children (**lc**) and an **lc*age** interaction:

```
▶ ADDT 'lc'
▶ ADDT 'lc' 'age'
```

Running this model produces:



$$\text{resp}_{ijk} \sim \text{Multinomial}(\text{cons}_{jk}, \pi_{ijk})$$

$$\log(\pi_{1jk} / \pi_{4jk}) = \beta_{0k}\text{cons.ster}_{ijk} + 0.077(0.034)\text{age.ster}_{ijk} + 2.083(0.366)\text{lc1.ster}_{ijk} + 2.297(0.359)\text{lc2.ster}_{ijk} + 2.559(0.352)\text{lc3plus.ster}_{ijk} + 0.025(0.040)\text{lc1.age.ster}_{ijk} + -0.022(0.039)\text{lc2.age.ster}_{ijk} + -0.105(0.036)\text{lc3plus.age.ster}_{ijk}$$

$$\beta_{0k} = -3.811(0.348) + v_{0k}$$

$$\log(\pi_{2jk} / \pi_{4jk}) = \beta_{1k}\text{cons.mod}_{ijk} + -0.035(0.019)\text{age.mod}_{ijk} + 1.072(0.243)\text{lc1.mod}_{ijk} + 1.041(0.233)\text{lc2.mod}_{ijk} + 1.163(0.226)\text{lc3plus.mod}_{ijk} + 0.033(0.027)\text{lc1.age.mod}_{ijk} + -0.027(0.028)\text{lc2.age.mod}_{ijk} + -0.053(0.023)\text{lc3plus.age.mod}_{ijk}$$

$$\beta_{1k} = -2.066(0.221) + v_{1k}$$

$$\log(\pi_{3jk} / \pi_{4jk}) = \beta_{2k}\text{cons.trad}_{ijk} + 0.008(0.024)\text{age.trad}_{ijk} + 0.751(0.318)\text{lc1.trad}_{ijk} + 1.087(0.287)\text{lc2.trad}_{ijk} + 1.227(0.278)\text{lc3plus.trad}_{ijk} + 0.005(0.037)\text{lc1.age.trad}_{ijk} + 0.004(0.033)\text{lc2.age.trad}_{ijk} + -0.021(0.028)\text{lc3plus.age.trad}_{ijk}$$

$$\beta_{2k} = -2.669(0.257) + v_{2k}$$

$$\begin{bmatrix} v_{0k} \\ v_{1k} \\ v_{2k} \end{bmatrix} \sim \text{N}(0,\ \Omega_v)\ :\ \Omega_v = \begin{bmatrix} 0.578(0.163) & & \\ 0.357(0.107) & 0.430(0.114) & \\ 0.253(0.101) & 0.165(0.083) & 0.332(0.112) \end{bmatrix}$$

$$\text{cov}(y_{sjk}, y_{rjk}) = -\ \pi_{sjk}\pi_{rjk}/\text{cons}_{jk} : s \neq r; \quad \pi_{sjk}(1 - \pi_{rjk})/\text{cons}_{jk} : s = r;$$

(8601 of 8601 cases in use)

Let's now set up a prediction to explore the interactions between age (old and young mothers), number of children and the probabilities of using different methods of contraception. In the **Customised predictions** window make the following specifications:

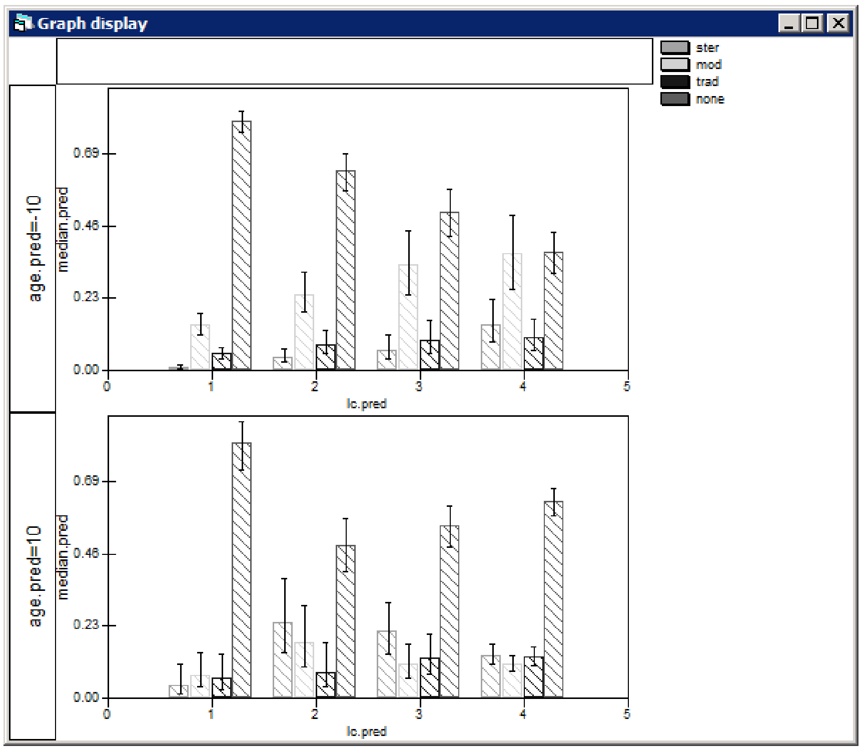
- **use4**: all categories
- **age**: **Values** (10, −10) (note you will have to remove the mean value of −0.3278697)
- **lc**: select all categories
- Select **Probabilities**
- Select **Medians**
- Click the **Fill grid**, **Predict** and **Plot Grid** buttons

and then set plot up as follows

- Select **Y:** median.pred, **X**: lc.pred,
- Select **Grouped by:** use4.pred
- Select **Trellis X**: age.pred
- Select **95% confidence intervals**
- Select **Error bars**

- Click **Apply**

This produces:



This graph shows that across ages (top graph panel = young, bottom graph panel = old) and parities (that is, number of children) the most prevelent contraceptive behaviour is to use no contraception at all. However, some interesting trends can be observed — for example, the probability of using no contaception is highest for women with no children. For both young and old women once they have had one child the probability of using some form of active contraception increases. For young women the pattern is more striking and the preferred method of active contraception for young women is modern contraception. Active contraceptive use is strongest for young women with 3 or more children.

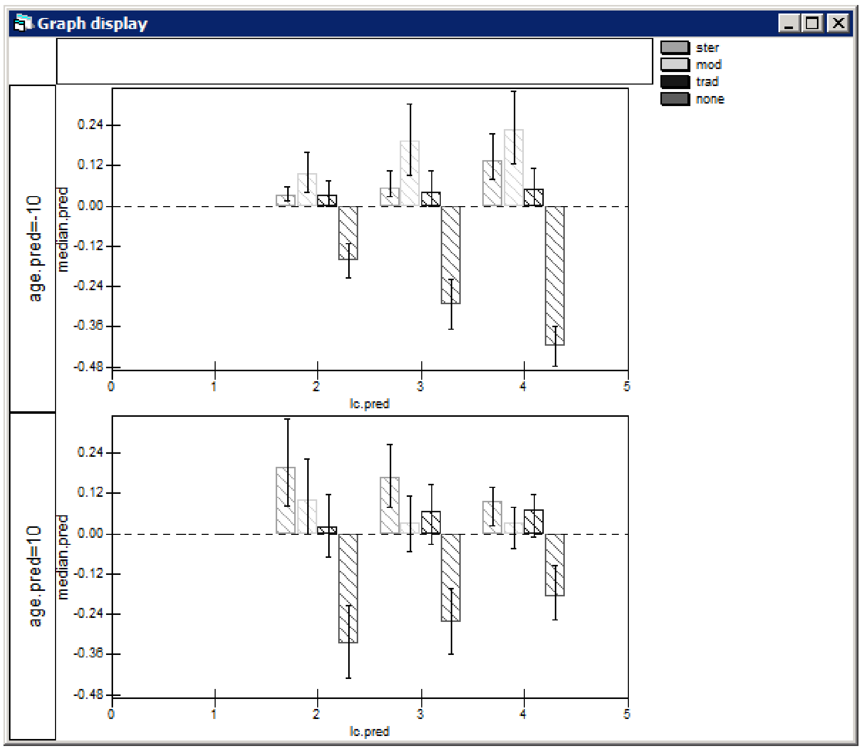
We may wish to explore whether differences from a particular reference group are statistically significant. For example, we may wish to choose women with no children as a reference group and evaluate:

$$p(\mathbf{usage} = \mathbf{ster}|\mathbf{lc} = \mathbf{lc1},\ \mathbf{age} = -10) - p(\mathbf{usage} = \mathbf{ster}|\mathbf{lc} = \mathbf{lc0},\ \mathbf{age} = -10)$$
$$\vdots$$
$$p(\mathbf{usage} = \mathbf{ster}|\mathbf{lc} = \mathbf{lc3},\ \mathbf{age} = -10) - p(\mathbf{usage} = \mathbf{ster}|\mathbf{lc} = \mathbf{lc0},\ \mathbf{age} = -10)$$

That is, see how the difference from women with no children changes as a function of age, usage type and number of children. In the **Customised predictions** window:

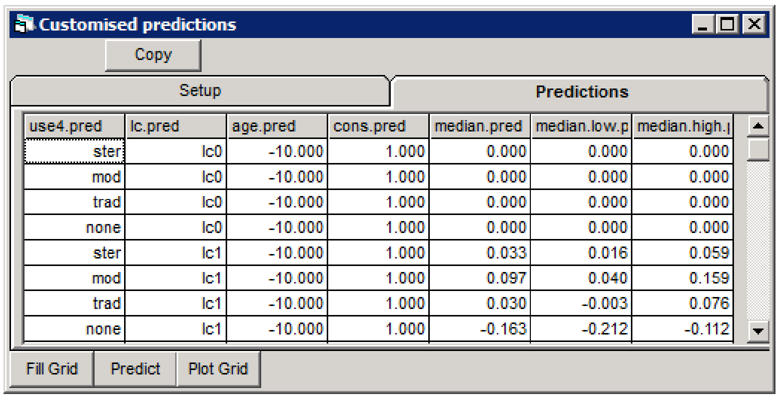
- Select **Differences**; **from variable**: lc; **Reference value**: lc0
- Click the **Predict** button

The graph display will update to:



Remember each bar represents the difference of a probability of usage for a specified method, age and number of children from the same method and age with number of children = 0. So all the bars corresponding to number of children = 0 disappear since we are subtracting a probability of usage for a particular combination of explanatory values from itself.

Let's look at the 4 bars for the 4 usage types at number of kids = 1, age = −10. That is, young women with one child. This is the leftmost cluster of bars in the top graph panel which corresponds to prediction cases 5-8 in the Predictions tab of the Customised predictions window.



use4.pred	lc.pred	age.pred	cons.pred	median.pred	median.low.p	median.high.p
ster	lc0	-10.000	1.000	0.000	0.000	0.000
mod	lc0	-10.000	1.000	0.000	0.000	0.000
trad	lc0	-10.000	1.000	0.000	0.000	0.000
none	lc0	-10.000	1.000	0.000	0.000	0.000
ster	lc1	-10.000	1.000	0.033	0.016	0.059
mod	lc1	-10.000	1.000	0.097	0.040	0.159
trad	lc1	-10.000	1.000	0.030	-0.003	0.076
none	lc1	-10.000	1.000	-0.163	-0.212	-0.112

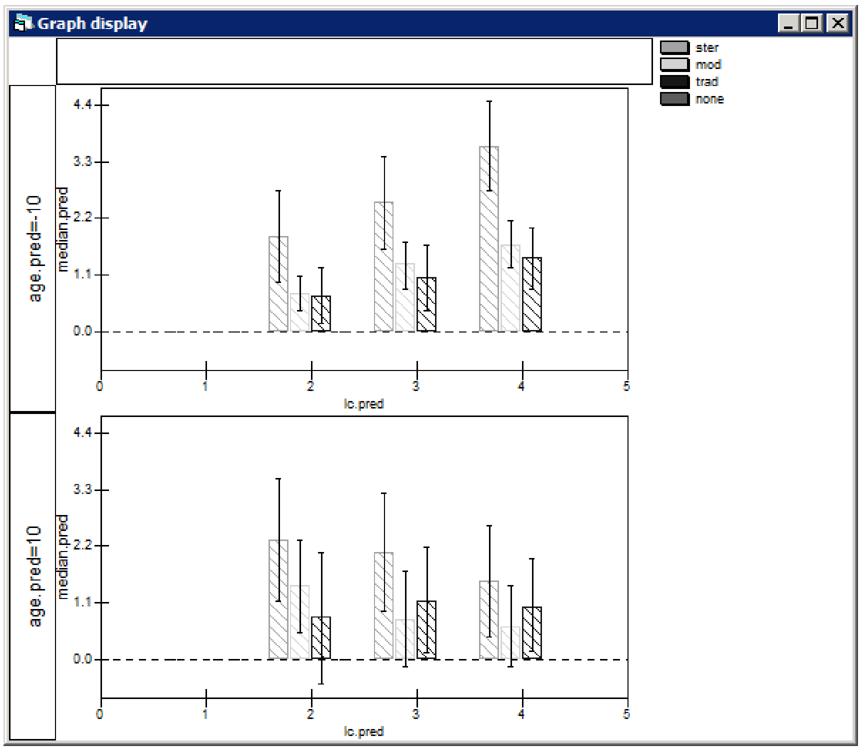
We see from row 8 of the above table (under median.pred) that

$$\begin{aligned} &p(\mathbf{method} = \mathbf{none}|\mathbf{lc} = \mathbf{lc1},\ \mathbf{age} = -10) \\ -&p(\mathbf{method} = \mathbf{none}|\mathbf{lc} = \mathbf{lc0},\ \mathbf{age} = -10) = -0.162 \end{aligned}$$

That is, the probability that young women with one child used no contraception is 0.162 less than young women with no children. Once young women have had a child they are more likely to use some active form of contraception. We can see which form of contraception these young women switch to by looking at rows 5–7 of the prediction table. We thus discover that 0.162 = 0.034(ster) + 0.097(mod) + 0.033(trad). Note that there is a tiny discrepancy in this equality due to rounding error.

We can carry out the same prediction on the logit scale and get log odds ratios. In the **Customised predictions** window

- Select **logit**
- Click **Predict**



Now the leftmost mid grey (blue on your screen) bar in the upper graph panel represents the log odds ratio

$$\log \left[\frac{p(\mathbf{usage} = \mathbf{ster}|\mathbf{age} = -10, \mathbf{lc} = \mathbf{lc1})/p(\mathbf{usage} = \mathbf{none}|\mathbf{age} = -10, \mathbf{lc} = \mathbf{lc1})}{p(\mathbf{usage} = \mathbf{ster}|\mathbf{age} = -10, \mathbf{lc} = \mathbf{lc0})/p(\mathbf{usage} = \mathbf{none}|\mathbf{age} = -10, \mathbf{lc} = \mathbf{lc0})} \right]$$

We get a log odds ratio (as opposed to a logit) because in our prediction we asked for logits to be differenced from lc = lc0.

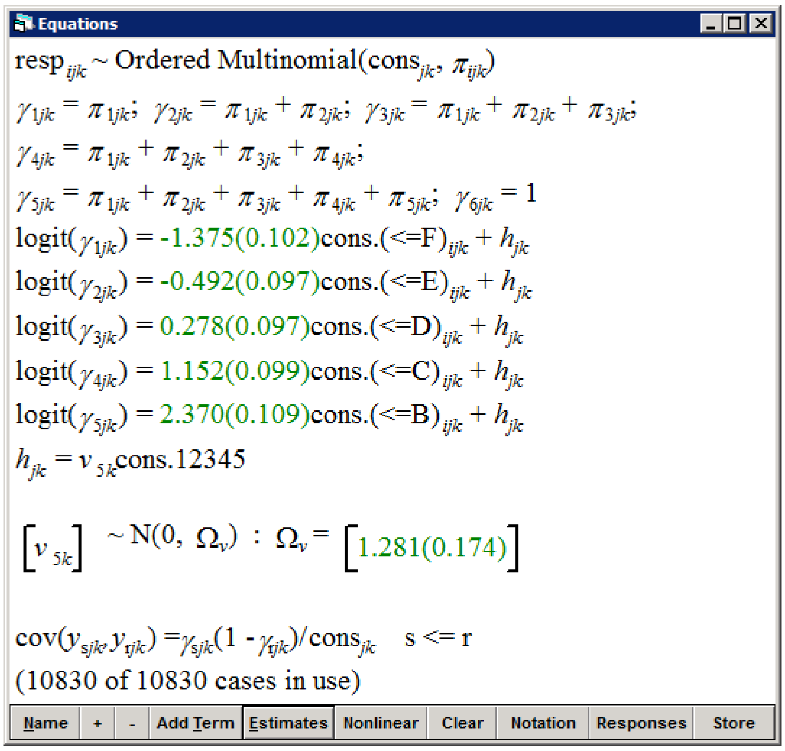
2.3.2 Ordered Multinomial

We will take the A-level dataset used in Chapter 11 of the User's Guide:

- Retrieve alevchem.ws

Let's set up a basic model using the following commands:

Command	Explanation
`MNOM 1 'a-point'c10 c11 6`	note the reference category is 6, <= A
`NAME c10 'resp'c11 'respcat'`	
`IDEN 1 'respcat' 2 'pupil' 3 'estab'`	specify level IDs
`ADDT 'CONS'`	add a different intercept for each response category
`RPAT 1 1 1 1 1`	specify that future added terms have same coefficient for each multinomial category
`ADDT 'cons'`	add a common intercept for categories 1–5
`SETV 3 'cons.12345'`	allow common intercept to vary across establishments
`FPAR 0 'cons.12345'`	remove common intercept from fixed part of the model so it is no longer overparameterized
`DOFFs 1 'cons'`	specify denominator
`LINEarise 1 2`	select PQL order 2



Equations

$\text{resp}_{ijk} \sim \text{Ordered Multinomial}(\text{cons}_{jk}, \pi_{ijk})$

$\gamma_{1jk} = \pi_{1jk};\ \gamma_{2jk} = \pi_{1jk} + \pi_{2jk};\ \gamma_{3jk} = \pi_{1jk} + \pi_{2jk} + \pi_{3jk};$

$\gamma_{4jk} = \pi_{1jk} + \pi_{2jk} + \pi_{3jk} + \pi_{4jk};$

$\gamma_{5jk} = \pi_{1jk} + \pi_{2jk} + \pi_{3jk} + \pi_{4jk} + \pi_{5jk};\ \gamma_{6jk} = 1$

$\text{logit}(\gamma_{1jk}) = -1.375(0.102)\text{cons.}(\text{<=F})_{ijk} + h_{jk}$

$\text{logit}(\gamma_{2jk}) = -0.492(0.097)\text{cons.}(\text{<=E})_{ijk} + h_{jk}$

$\text{logit}(\gamma_{3jk}) = 0.278(0.097)\text{cons.}(\text{<=D})_{ijk} + h_{jk}$

$\text{logit}(\gamma_{4jk}) = 1.152(0.099)\text{cons.}(\text{<=C})_{ijk} + h_{jk}$

$\text{logit}(\gamma_{5jk}) = 2.370(0.109)\text{cons.}(\text{<=B})_{ijk} + h_{jk}$

$h_{jk} = v_{5k}\text{cons.12345}$

$$\begin{bmatrix} v_{5k} \end{bmatrix} \sim \text{N}(0,\ \Omega_v)\ :\ \Omega_v = \begin{bmatrix} 1.281(0.174) \end{bmatrix}$$

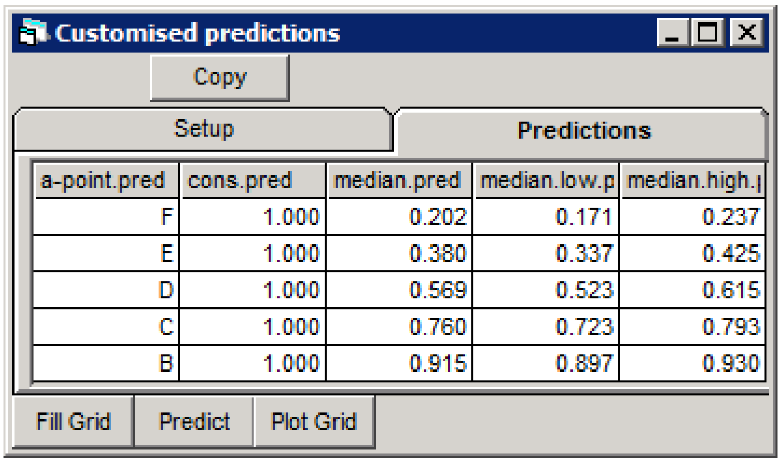
$\text{cov}(y_{sjk}, y_{rjk}) = \gamma_{sjk}(1 - \gamma_{rjk})/\text{cons}_{jk} \quad s <= r$

(10830 of 10830 cases in use)

In the **Customised predictions** window:

- Set **a-point** to all categories *apart* from the reference category A
- Select **Probabilities**
- Select **Medians**
- Click the **Fill grid** and **Predict** buttons
- Select the **Predictions** tab

This prediction gives the cumulative probabilities of passing across the grade categories



Customised predictions

a-point.pred	cons.pred	median.pred	median.low.p	median.high.|
F	1.000	0.202	0.171	0.237
E	1.000	0.380	0.337	0.425
D	1.000	0.569	0.523	0.615
C	1.000	0.760	0.723	0.793
B	1.000	0.915	0.897	0.930

The **Differences from** functionality, in the **Customised predictions** window, is not currently implemented for ordered multinomial models.

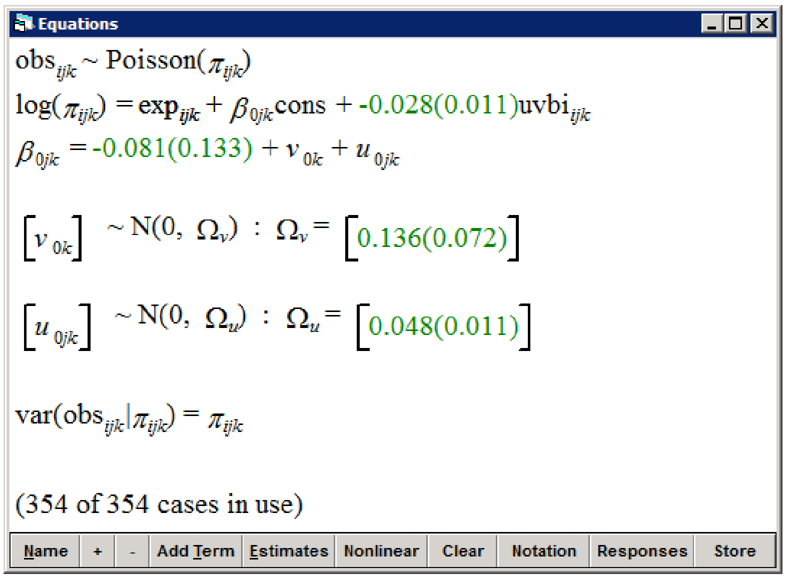
2.4 Poisson models

We will work with the skin cancer dataset used in Chapter 12 of the User's Guide.

- Open the worksheet **mmmec.ws**

Let's model malignant melanoma county level death rates as a function of UVBI exposure, allowing for between region and country variation:

Command	Explanation
`resp 'obs'`	
`iden 1 'county' 2 'region' 3 'nation'`	
`expl 1 'cons' 'uvbi'`	
`rdist 1 1`	set response to be Poisson
`lfun 3`	specify log link function
`doffs 1 'exp'`	offsets in 'exp'
`loge 'exp' 'exp'`	
`setv 2 'cons'`	
`setv 3 'cons'`	
`linea 1 2`	



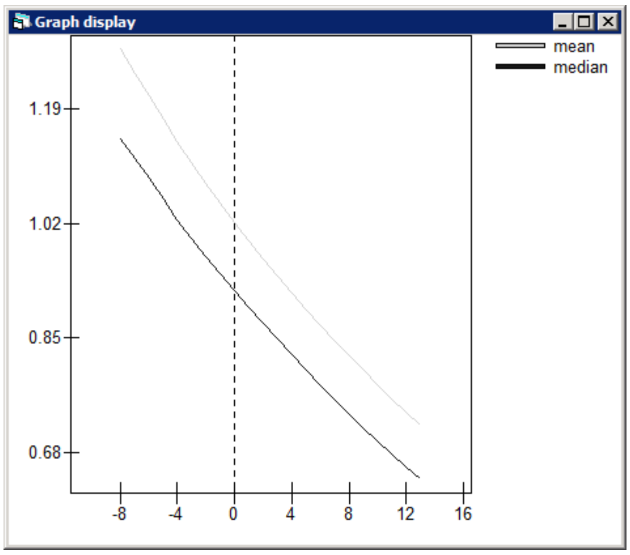
$$\text{obs}_{ijk} \sim \text{Poisson}(\pi_{ijk})$$
$$\log(\pi_{ijk}) = \text{exp}_{ijk} + \beta_{0jk}\text{cons} + -0.028(0.011)\text{uvbi}_{ijk}$$
$$\beta_{0jk} = -0.081(0.133) + v_{0k} + u_{0jk}$$
$$\left[v_{0k}\right] \sim \text{N}(0,\ \Omega_v)\ :\ \Omega_v = \left[0.136(0.072)\right]$$
$$\left[u_{0jk}\right] \sim \text{N}(0,\ \Omega_u)\ :\ \Omega_u = \left[0.048(0.011)\right]$$
$$\text{var}(\text{obs}_{ijk}|\pi_{ijk}) = \pi_{ijk}$$

(354 of 354 cases in use)

In the **Customised predictions** window

- Set **uvbi** to range {13,−8, 1}
- Select **Medians** and **Means**

- Click the **Fill Grid** and **Predict** buttons
- Using the **Customised graphs** window from the Graphs menu, plot the median and the mean predictions (**median.pred** and **mean.pred**) against UBVI exposure (**uvbi.pred**)

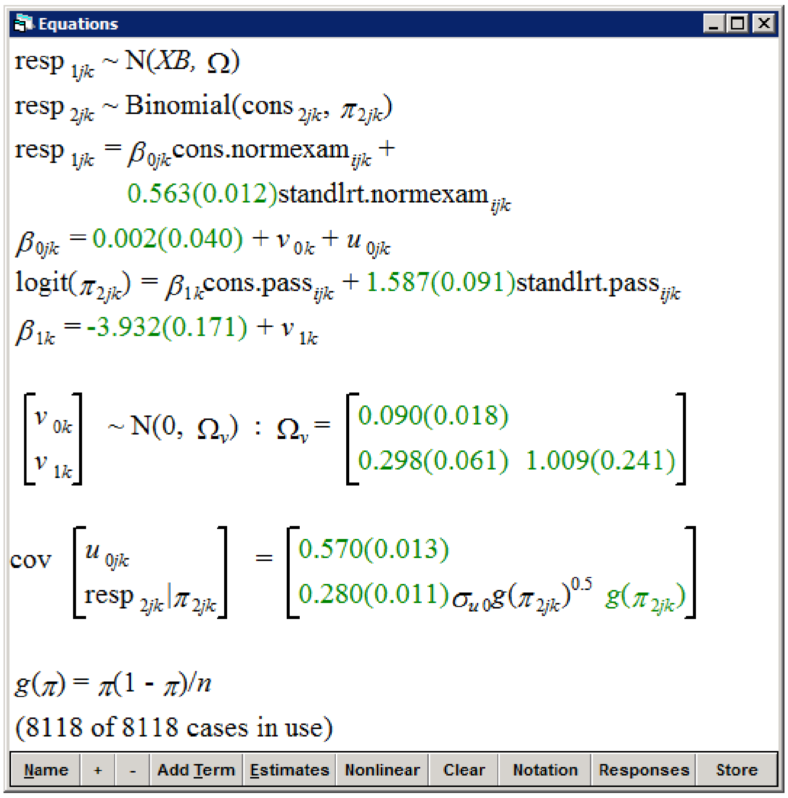


We see that the mean is uniformly higher than the median. This is due to the shape of the link function. This difference that we see emphasises the need to consider the shape of the distribution when reporting results or making inferences: for example, just as we saw for binomial models, the mean probability cannot be obtained by taking the exponential of the mean XB; and confidence intervals must be found before, not after, transforming to probabilities via the exponential. This also explains why we won't expect the predicted value of, for example, the unconditional mean to exactly match our observed value.

2.5 Multivariate models

The **Customised predictions** window can deal with only one response at a time when a multivariate model is set up. To demonstrate this, let's create a binary variable from the exam scores in the tutorial data. We then fit a multilevel bivariate response model with the original continuous score as the first response and the dichotomised variable as the second response. Open the **tutorial** worksheet and type the following commands

```
▶ calc c11 = c3 > 1.5
▶ name c11 'pass'
▶ mvar 1 c3 c11
▶ expl 1 'cons' 'standlrt'
▶ iden 2 'student' 3 'school'
▶ setv 3 'cons.normexam' 'cons.pass'
▶ setv 2 'cons.normexam'
▶ linea 1 2
▶ rdist 2 0
▶ doffs 2 'cons'
```



$$\mathrm{resp}_{1jk} \sim \mathrm{N}(XB,\ \Omega)$$

$$\mathrm{resp}_{2jk} \sim \mathrm{Binomial}(\mathrm{cons}_{2jk},\ \pi_{2jk})$$

$$\mathrm{resp}_{1jk} = \beta_{0jk}\mathrm{cons.normexam}_{ijk} + 0.563(0.012)\mathrm{standlrt.normexam}_{ijk}$$

$$\beta_{0jk} = 0.002(0.040) + v_{0k} + u_{0jk}$$

$$\mathrm{logit}(\pi_{2jk}) = \beta_{1k}\mathrm{cons.pass}_{ijk} + 1.587(0.091)\mathrm{standlrt.pass}_{ijk}$$

$$\beta_{1k} = \text{-}3.932(0.171) + v_{1k}$$

$$\begin{bmatrix} v_{0k} \\ v_{1k} \end{bmatrix} \sim \mathrm{N}(0,\ \Omega_v)\ :\ \Omega_v = \begin{bmatrix} 0.090(0.018) & \\ 0.298(0.061) & 1.009(0.241) \end{bmatrix}$$

$$\mathrm{cov}\begin{bmatrix} u_{0jk} \\ \mathrm{resp}_{2jk}|\pi_{2jk} \end{bmatrix} = \begin{bmatrix} 0.570(0.013) & \\ 0.280(0.011)\sigma_{u0}g(\pi_{2jk})^{0.5} & g(\pi_{2jk}) \end{bmatrix}$$

$$g(\pi) = \pi(1 - \pi)/n$$

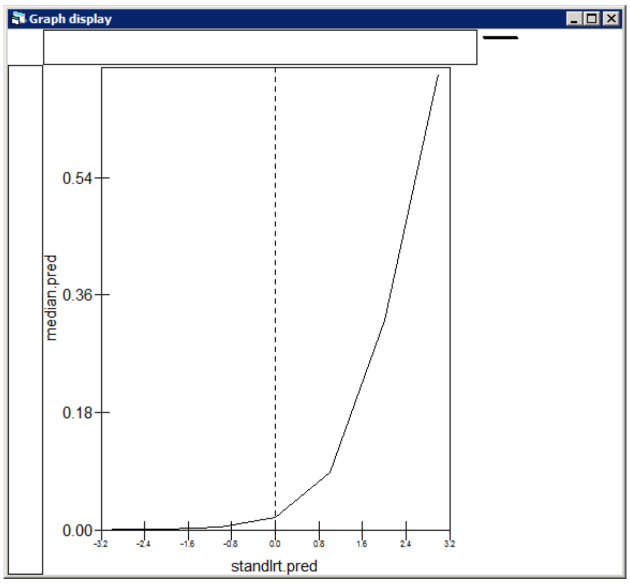
(8118 of 8118 cases in use)

In the **Customised prediction** window notice in the top right hand corner the name of the response we are working with, **normexam**. This means that any prediction we specify will be applied to the continuous response **normexam** only.

- Set **standlrt** to range {3,−3, 1}
- Click **Fill Grid**, **Predict and Plot Grid**
- Set **X:** standlrt.pred, **Y:** mean.pred
- Click **Apply**

Now let's make a prediction for the binomial response. In the **Customised predictions** window

- Change **normexam** to pass in the drop-down list on the top right of the window
- Click **Predict** and **Plot Grid**
- Set **Y**: median.pred and Click **Apply**



3 3D Graphics

We have implemented some preliminary 3D graphics in the software. These are currently implemented as 3 commands:

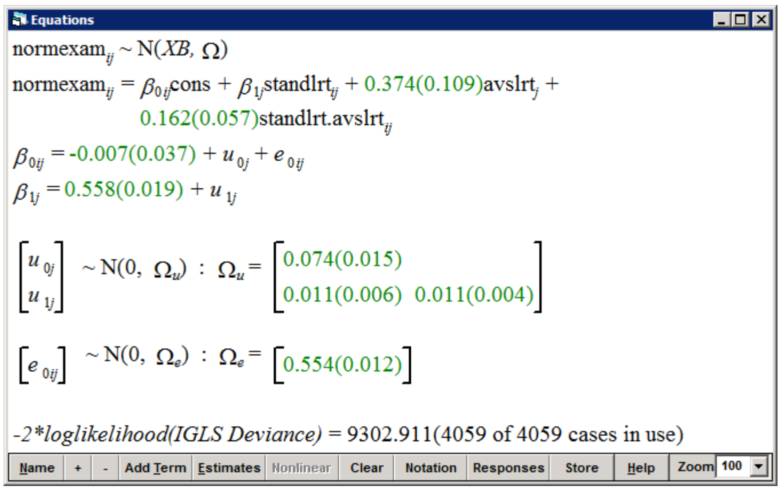
SURF 3d graph number **N**, **X**, **Y**, **Z**

SCATt 3d graph number **N**, **X**, **Y**, **Z**, [group column]

SHOW 3d graph number **N**

Let's start by constructing a surface with the Customised predictions window and then plotting it.

Return to the model



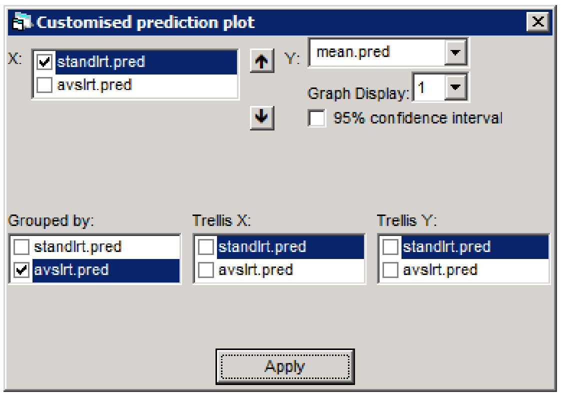
$$\text{normexam}_{ij} \sim \text{N}(XB, \Omega)$$
$$\text{normexam}_{ij} = \beta_{0ij}\text{cons} + \beta_{1j}\text{standlrt}_{ij} + 0.374(0.109)\text{avslrt}_j + 0.162(0.057)\text{standlrt.avslrt}_{ij}$$
$$\beta_{0ij} = -0.007(0.037) + u_{0j} + e_{0ij}$$
$$\beta_{1j} = 0.558(0.019) + u_{1j}$$
$$\begin{bmatrix} u_{0j} \\ u_{1j} \end{bmatrix} \sim \text{N}(0, \Omega_u) \ : \ \Omega_u = \begin{bmatrix} 0.074(0.015) & \\ 0.011(0.006) & 0.011(0.004) \end{bmatrix}$$
$$\begin{bmatrix} e_{0ij} \end{bmatrix} \sim \text{N}(0, \Omega_e) \ : \ \Omega_e = \begin{bmatrix} 0.554(0.012) \end{bmatrix}$$

*-2*loglikelihood(IGLS Deviance)* = 9302.911(4059 of 4059 cases in use)

In the **Customised Predictions** window

- Set the range for **standlrt** to be Upper Bound: 3, Lower: −3, Increment: 1
- Set the range for **avslrt** to be Upper Bound: 0.5, Lower: −0.5, Increment: 0.1
- Click **Fill Grid** and **Predict**

Set up the plot using



Customised prediction plot

X: [✓ standlrt.pred] avslrt.pred

Y: mean.pred

Graph Display: 1

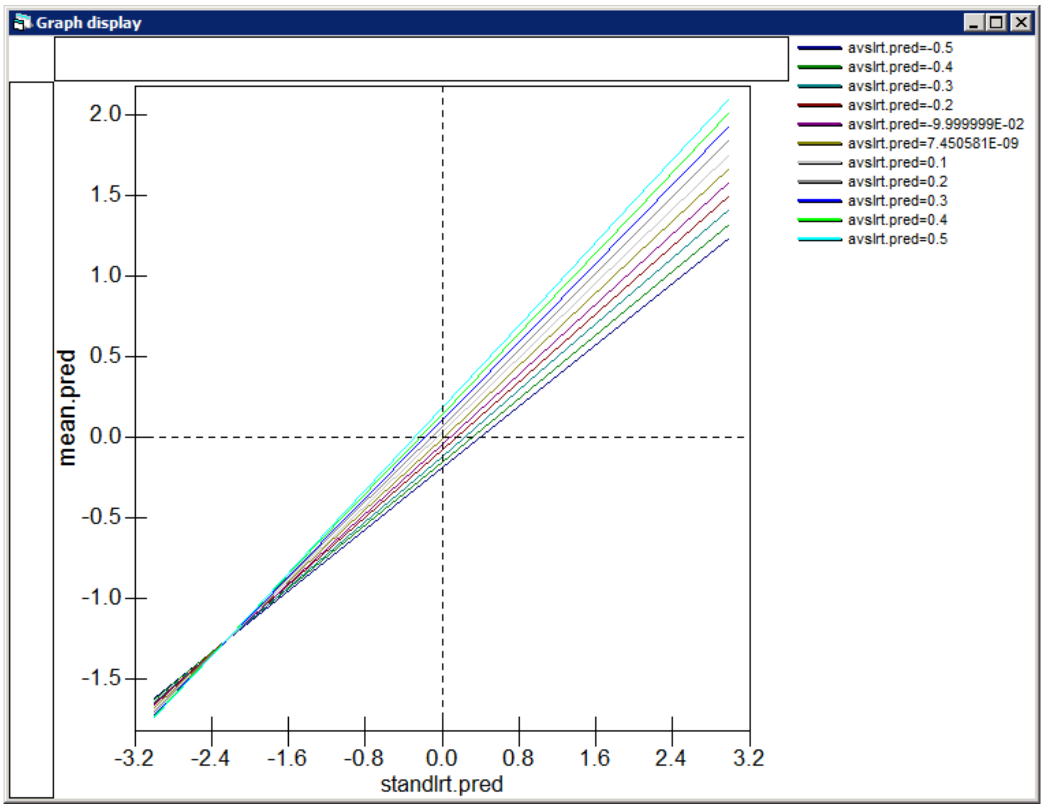
95% confidence interval

Grouped by: standlrt.pred [✓ avslrt.pred]

Trellis X: standlrt.pred avslrt.pred

Trellis Y: standlrt.pred avslrt.pred

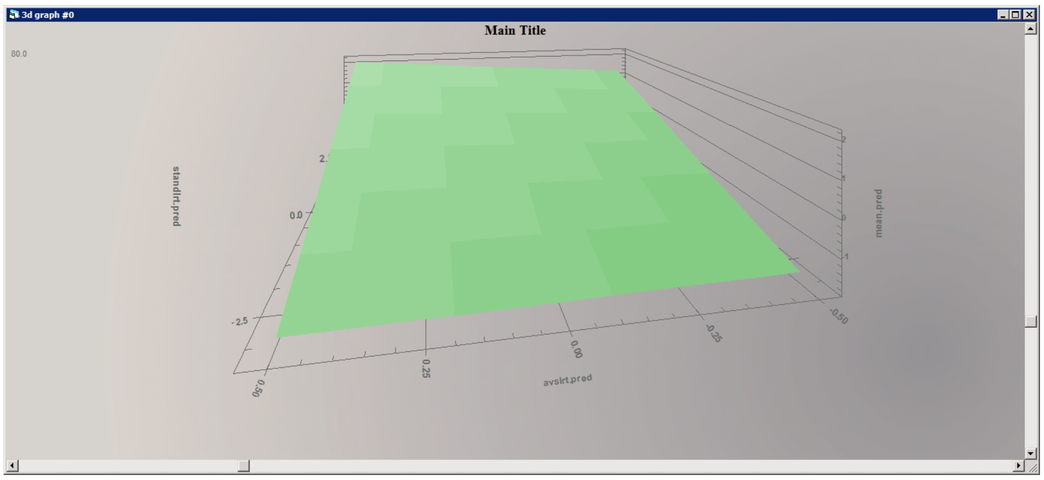
which produces:



To plot this as a surface, in the **Command interface** window type

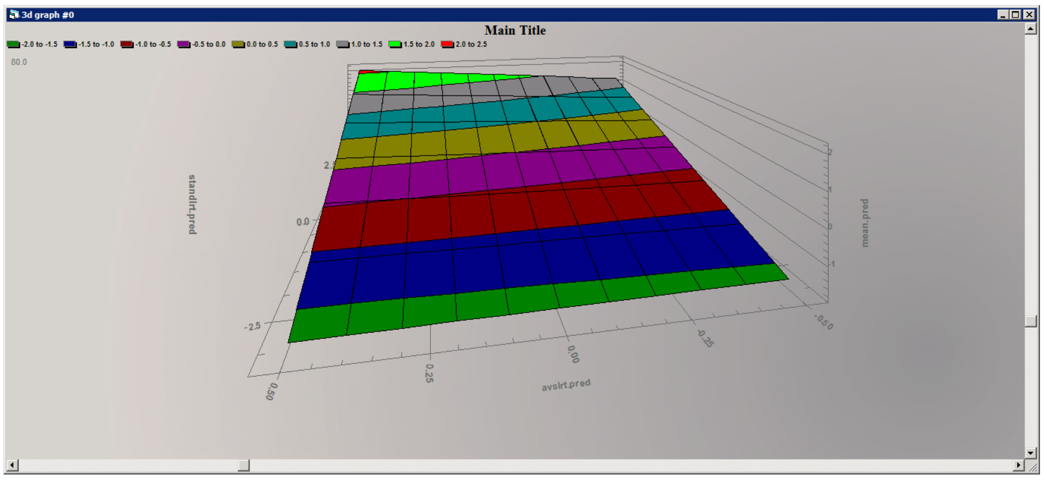
```
▶ surf 1 'standlrt.pred' 'avslrt.pred' 'mean.pred'
▶ show 1
```

which produces (after rotating the view using the horizontal slider to 80.0 — see top left of window for current view angle):



Right clicking on the graph and selecting

<Plotting Method><Surface with Contouring> produces



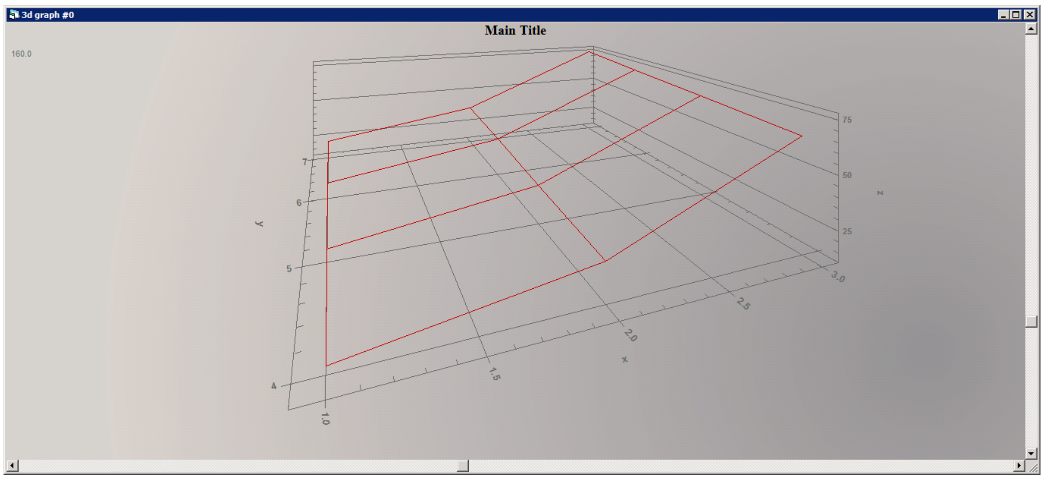
The SURF plot requires a rectangular data grid of X, Y values with a Z value in each cell of the grid. So to plot the function $z = 2x^3 + 3y$ for $x = \{1, 2, 3\}$, $y = \{4, 5, 6, 7\}$ we need to first construct

x	y
1	4
1	5
1	6
1	7
⋮	⋮
3	4
3	5
3	6
3	7

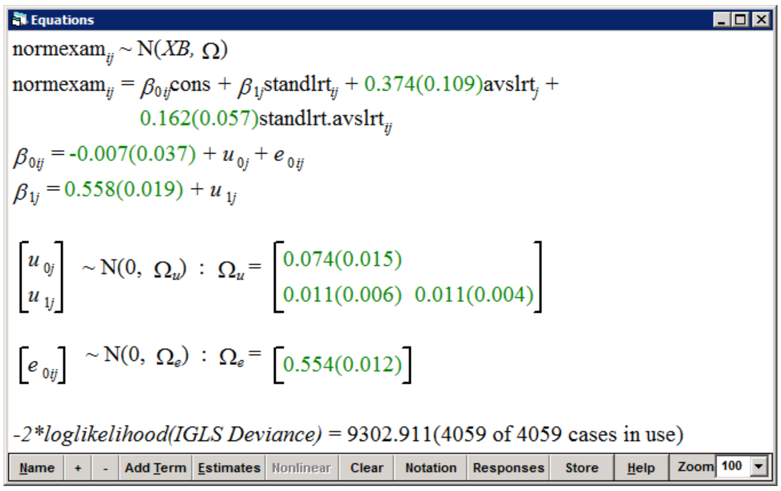
then create z from these two columns. This can be done by typing the following commands into the **Command interface** window (close the 3d graph display window first)

```
▶ gene 1 3 1 c100
▶ gene 4 7 1 c101
▶ ucom c100 c101 c102 c103
▶ name c102 'x' c103 'y' c104 'z'
▶ calc 'z'=2*'x'^3+3*'y'
▶ surf 1 'x' 'y' 'z'
▶ show 1
```

which produces (after right clicking on the graph and selecting <Plotting Method><Wire Frame>):



Going back to the model we have just fitted:


Equations

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

$\text{normexam}_{ij} = \beta_{0ij}\text{cons} + \beta_{1j}\text{standlrt}_{ij} + 0.374(0.109)\text{avslrt}_{j} + 0.162(0.057)\text{standlrt.avslrt}_{ij}$

$\beta_{0ij} = -0.007(0.037) + u_{0j} + e_{0ij}$

$\beta_{1j} = 0.558(0.019) + u_{1j}$

$$\begin{bmatrix} u_{0j} \\ u_{1j} \end{bmatrix} \sim \text{N}(0,\ \Omega_u)\ :\ \Omega_u = \begin{bmatrix} 0.074(0.015) & \\ 0.011(0.006) & 0.011(0.004) \end{bmatrix}$$

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

*-2*loglikelihood(IGLS Deviance)* = 9302.911(4059 of 4059 cases in use)

the intercept and slopes are distributed

$$\theta = \begin{bmatrix} u_{0j} \\ u_{1j} \end{bmatrix} \sim N(0, \Omega_u), \qquad \Omega_u = \begin{bmatrix} 0.074 & \\ 0.011 & 0.011 \end{bmatrix}$$

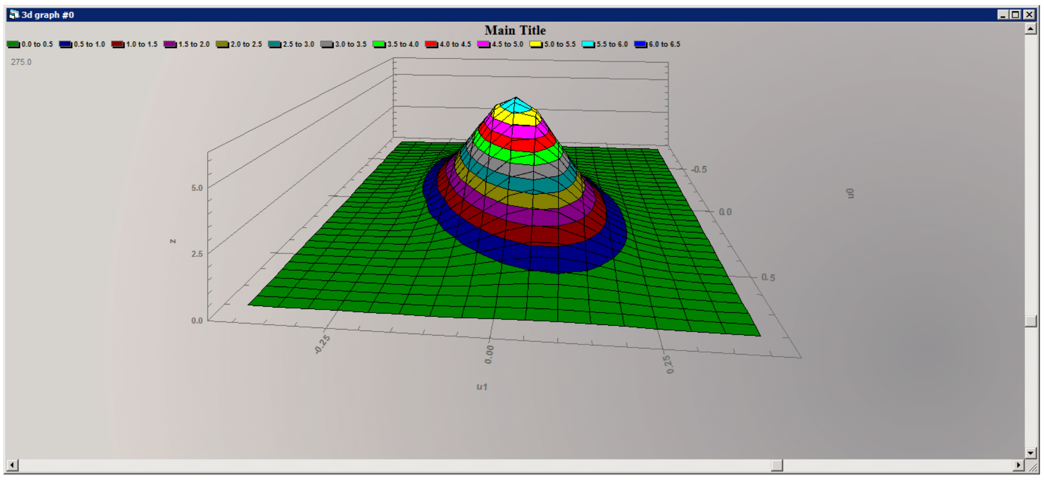
The probability density function for this bivariate Normal model where the means of the two variables are 0 is

$$p(\theta) = \frac{1}{(2\pi)}|\Omega_u|^{-1/2} \exp(-\frac{1}{2}(\theta^T \Omega_u^{-1} \theta)$$

To calculate and plot this bivariate distribution, close the graph then type:

Command	**Explanation**
`erase 'z'`	
`join 0.074 0.011 0.011 c100`	create Ω_u
`calc c101 = sym(c100)`	fill in (1,2) value of Ω_u(omitted in notation because Ω_u a symmetric matrix)
`gene -0.8 0.8 0.05 c102`	generate values for u_{0j}
`gene -0.4 0.4 0.05 c103`	generate values for u_{1j}
`ucom c102 c103 c104 c105`	create all combinations of (u_{0j}, u_{1j})
`name c104 'u0' c105 'u1' c107 'z'`	
`count 'u1' b1`	
`join 'u0' 'u1' c106`	
`matr c106 b1 2`	turn this grid into a $b1$ by 2 matrix
`calc 'z' = 1/(2*3.1416) * det(c101)^(-0.5) * expo(-0.5*diag(c106 *.inv(c101) *.(~c106)))`	Apply the Normal PDF to all values of of (u_{0j}, u_{1j}) in c106
`surf 1 'u0' 'u1' 'z'`	
`show 1`	

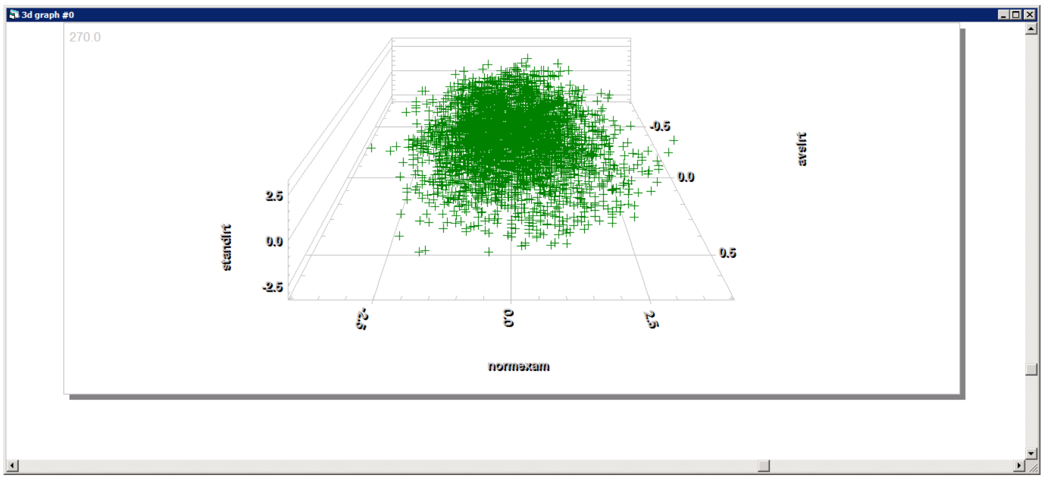
Selecting <plotting method><surface with contouring> on the graph that appears produces



The SCATter command can produce 3D scatters. For example,

- Close the **3d graph** window
- In the **Command interface** window type:

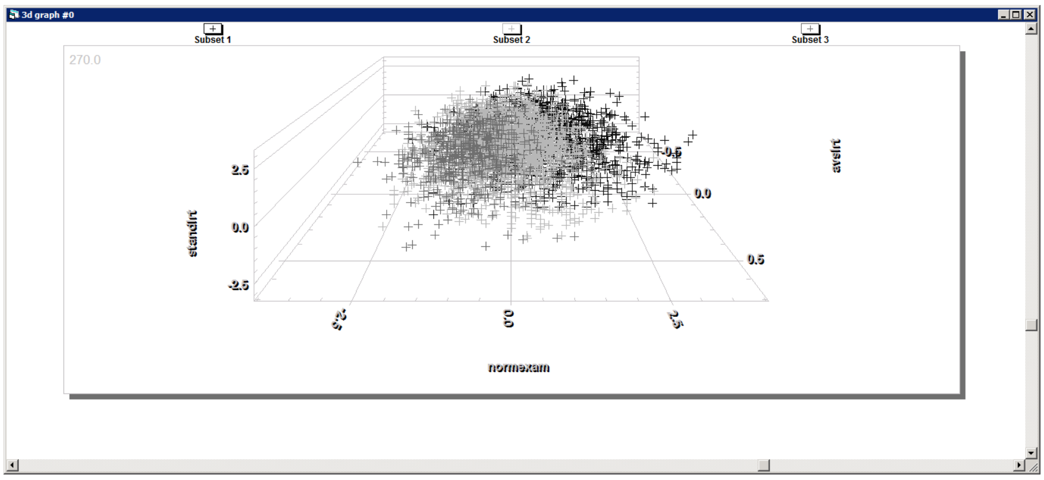
```
▶ SCAT 1 'avslrt' 'normexam' 'standlrt'
▶ Show 1
```



We can group the plot by vrband:

- Close the **3D graph window**
- In the **Command interface** window type:

```
▶ SCAT 1 'avslrt' 'normexam' 'standlrt' 'vrband'
▶ show 1
```



4 Model Comparison Tables

MLwiN now has a mechanism for storing the results of a series of models and displaying them in a single table. This has two main uses. Firstly, it is useful when conducting an analysis as it enables the user to easily see the results of a series of models. Secondly, often when writing papers, results from a series of models are presented in a single table; constructing these tables manually can be a time consuming and error prone process. To a large extent MLwiN now automates this process.

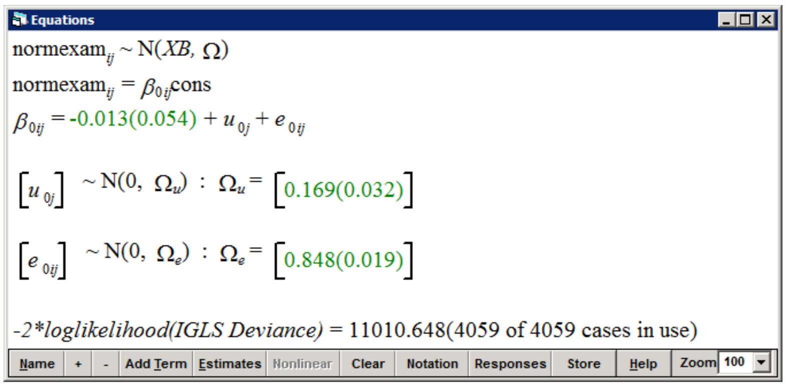
Let's set up a variance components model on the **tutorial** dataset

- Open the **tutorial** worksheet
- In the **Equations** window click on **y**
- In the **y variable** window select:

y:	Normexam
N levels:	2
level 2(j):	school
level 1(i):	student

- Click **done**
- Click on $\beta_0 x_0$
- Select **cons** from the drop-down list in the **X variable** window
- Tick **j(school), i(student)**
- Click **Done**

Running the model gives:



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

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

$\beta_{0ij} = -0.013(0.054) + u_{0j} + e_{0ij}$

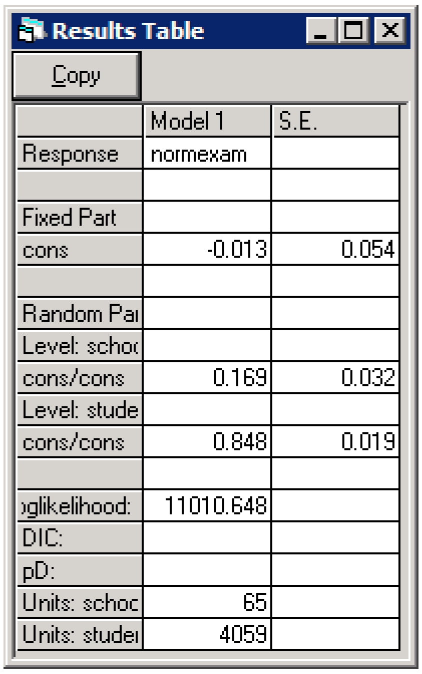
$\left[u_{0j}\right] \sim \text{N}(0, \Omega_u) : \Omega_u = \left[0.169(0.032)\right]$

$\left[e_{0ij}\right] \sim \text{N}(0, \Omega_e) : \Omega_e = \left[0.848(0.019)\right]$

*-2*loglikelihood(IGLS Deviance)* = 11010.648(4059 of 4059 cases in use)

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

We can add this model to the stored sequence of models under the name 'Model 1' by clicking on the **Store** button at the bottom of the **Equations** window and typing a name for the model in the window which appears, in this case **Model 1**. We can view the sequence of stored models (so far only one model) by selecting **Compare stored models** from the **Model** menu. This produces:


Results Table

Copy

	Model 1	S.E.
Response	normexam	
Fixed Part		
cons	-0.013	0.054
Random Pa		
Level: schoo		
cons/cons	0.169	0.032
Level: stude		
cons/cons	0.848	0.019
oglikelihood:	11010.648	
DIC:		
pD:		
Units: schoo	65	
Units: studei	4059	

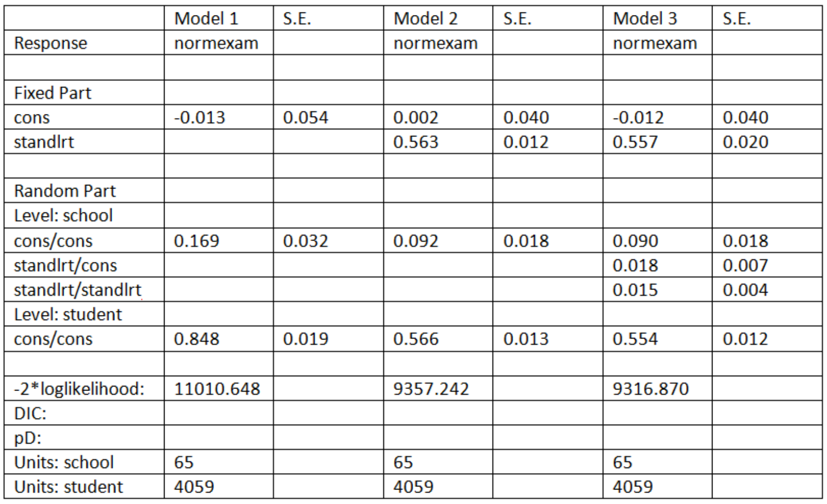
Clicking the **Copy** button will paste a tab-delimited text file into the clipboard.

- Click the **Copy** button

In Microsoft Word:

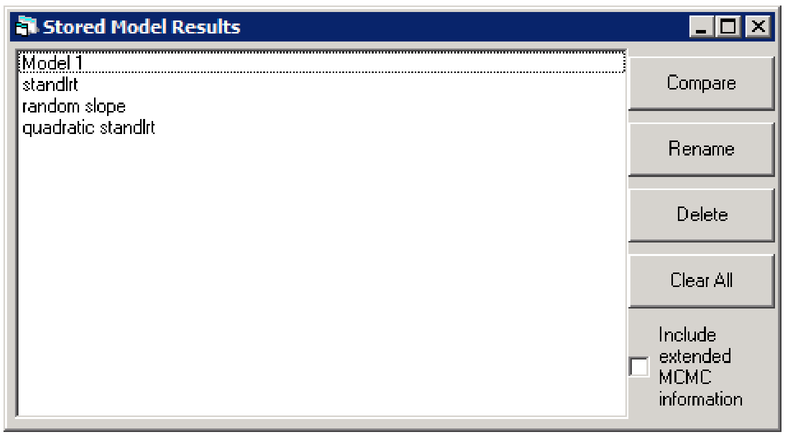
- Paste the text into a document
- Highlight the pasted text in Microsoft Word
- Select the <Table><Insert><Table. . . > menu item

This produces:



	Model 1	S.E.	Model 2	S.E.	Model 3	S.E.
Response	normexam		normexam		normexam	
Fixed Part						
cons	-0.013	0.054	0.002	0.040	-0.012	0.040
standlrt			0.563	0.012	0.557	0.020
Random Part						
Level: school						
cons/cons	0.169	0.032	0.092	0.018	0.090	0.018
standlrt/cons					0.018	0.007
standlrt/standlrt					0.015	0.004
Level: student						
cons/cons	0.848	0.019	0.566	0.013	0.554	0.012
-2*loglikelihood:	11010.648		9357.242		9316.870	
DIC:						
pD:						
Units: school	65		65		65	
Units: student	4059		4059		4059	

We can see a list of all models currently stored by selecting **Manage stored models** from the **Model** menu (shown here after storing several more models).



This allows us to select just some of the models to view in a results table, rather than having to display all of them as we would if we selected **Compare stored models**; to delete any or all stored models; or to rename any stored model. Note that there is no way to bring any of the stored models up again in the Equations window - if you wish to be able to easily return to working with a certain model after moving on to another one, you should save the worksheet (probably under a different name), and then you can return to that model by returning to that saved worksheet.

The tick-box labelled **Include extended MCMC information** controls how much information is displayed for models estimated using MCMC. Regardless of whether this box is ticked or not, when the model is stored, as well as all the same information that is stored for models estimated using (R)IGLS, the median of the chain for each parameter, upper and lower ends of the 95% credible interval, and effective sample size are also stored. This information is only displayed in the Results Table if the **Include extended MCMC information** box is ticked.

Note that neither the Results Table nor the list of stored models will not refresh automatically when a new model is stored. You will need to close them and once again select **Compare stored models** or **Manage stored models** from the **Model** menu for the new model to be included.

Storing and retrieving of model results can also be done using commands. The formats of the model table commands are

`MSTO <S>` : store model results as **S**

for example, entering the command

```
▶ MSTOre 'modela'
```

appends the current model to the table of models and names the model ‘modela’

`MPRI` : print stored model results to Output window

MCOM <**S**> ... <**S**> : compare the listed models; compare all if no parameters

for example, entering the command

```
▶ MCOM 'Model 1' 'modela' 'Model 4'
```

would create a model table comparing Model 1, modela and Model 4

`MERA` <**S**> : erase stored model results for listed models

for example:

```
▶ MERA 'modela'
```

erases modela

`MWIPe` : erase all stored models

Let’s create a macro to run a sequence of models and store each one in a model comparison table

Command	**Explanation**
`clear`	clear any current model
`mwipe`	clear model comparison table
`resp 'normexam'`	set response
`iden 1 'student' 2 'school'`	set level IDs
`addt 'cons'`	intercept
`setv 2`	set variance components
`setv 1`	
`expa 3`	expand terms in Equations window
`estm 2`	show estimates in Equations window
`maxi 1000`	set maximum iterations to 1000
`batc 1`	do not pause in between iterations
`star 1`	run the variance components model

`msto 'Model 1'`	store the results
`addt 'standlrt'`	add a slope to the model
`next 1`	run the new model
`msto 'Model 2'`	store the fixed slope variance components model
`setv 2`	add a random slope
`next 1`	run the random slopes model
`msto 'Model 3'`	store the random slopes model

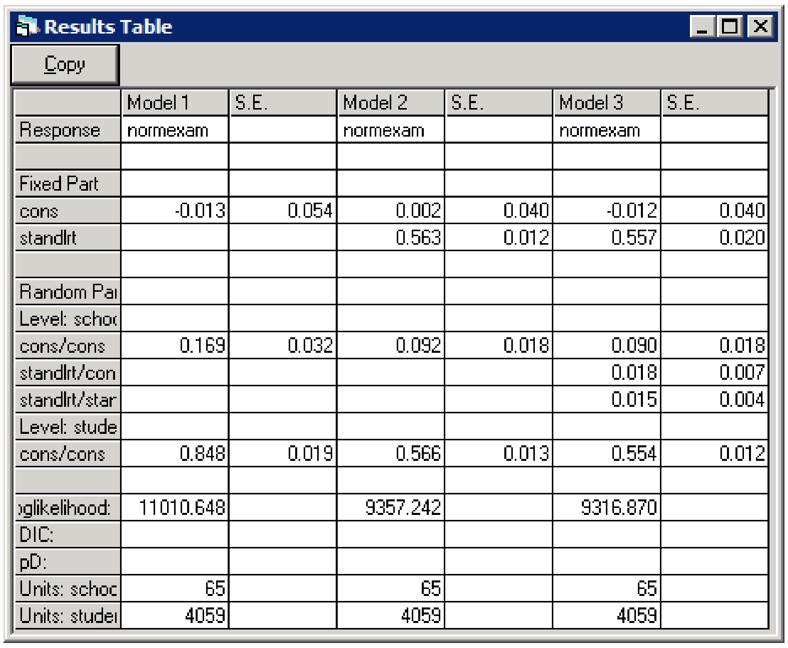
Note the **START** and **NEXT** commands when called from a macro with parameter 1 will cause the **Equations** window to be updated after each modelling iteration. Sometimes this can result in the software spending too much time updating the screen and not enough time doing sums. In this case you may prefer to use the **START** and **NEXT** commands with no added parameter, in which case screens are not updated during macro file execution. You can however place a **PAUSE 1** command at any point in a macro script which will cause all displayed windows to update themselves.

- From the **File** menu select **New Macro**
- Type or paste the above command sequence into the macro window
- Make sure the **Equations** window is open and visible
- Click the **Execute** button at the bottom of the macro window

You should now see the sequence of requested models being executed in the **Equations** window. To view the model comparison table, type

```
▶ mcomp
```

in the **Command interface** window or select **Compare stored models** from the **Model** menu. This produces:



	Model 1	S.E.	Model 2	S.E.	Model 3	S.E.
Response	normexam		normexam		normexam	
Fixed Part						
cons	-0.013	0.054	0.002	0.040	-0.012	0.040
standlrt			0.563	0.012	0.557	0.020
Random Pa						
Level: schoo						
cons/cons	0.169	0.032	0.092	0.018	0.090	0.018
standlrt/con					0.018	0.007
standlrt/star					0.015	0.004
Level: stude						
cons/cons	0.848	0.019	0.566	0.013	0.554	0.012
glikelihood:	11010.648		9357.242		9316.870	
DIC:						
pD:						
Units: schoo	65		65		65	
Units: stude	4059		4059		4059	

Suppose we decided to recode a variable, e.g., turning **standlrt** into a binary variable. We can recode the variable and reanalyse and the model comparison table will get reformed.

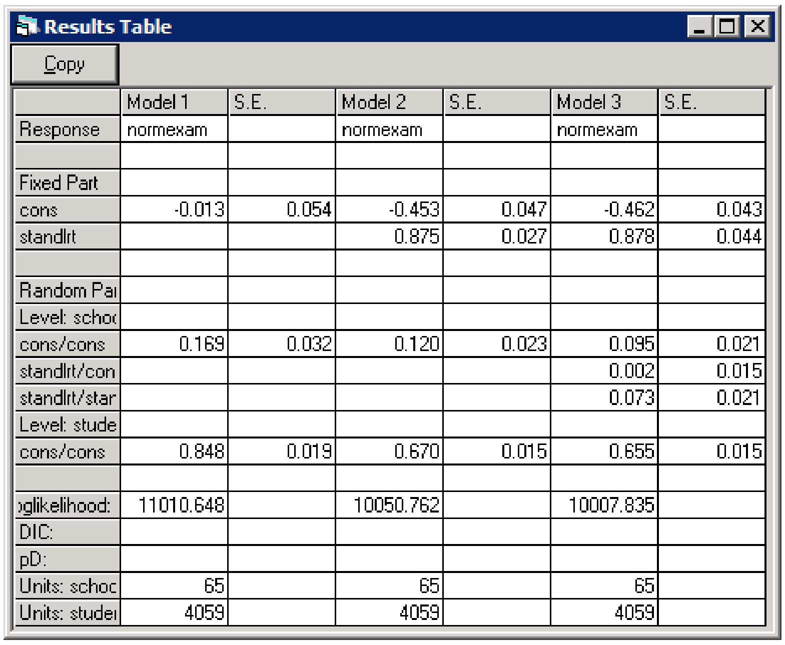
- In the **Command interface** window, type

```
▶ calc 'standlrt' = 'standlrt' > 0
```

- Rerun the analysis macro file
- In the **Command interface** window, type

```
▶ mcomp
```

which then produces the updated results table:



Results Table

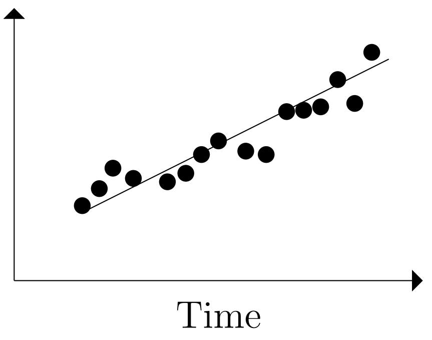
Copy

	Model 1	S.E.	Model 2	S.E.	Model 3	S.E.
Response	normexam		normexam		normexam	
Fixed Part						
cons	-0.013	0.054	-0.453	0.047	-0.462	0.043
standlrt			0.875	0.027	0.878	0.044
Random Pa						
Level: schoo						
cons/cons	0.169	0.032	0.120	0.023	0.095	0.021
standlrt/con					0.002	0.015
standlrt/star					0.073	0.021
Level: stude						
cons/cons	0.848	0.019	0.670	0.015	0.655	0.015
glikelihood:	11010.648		10050.762		10007.835	
DIC:						
pD:						
Units: schoo	65		65		65	
Units: stude	4059		4059		4059	

5 A new method for estimating autocorrelated errors in continuous time

Previous versions of MLwiN implemented in a set of macro files the algorithms in Goldstein *et al.* (1994) to estimate models with autocorrelated errors at level 1. The macros were rather unstable and we removed them in version 2.02 of MLwiN. We introduce here a simpler method of estimating these models.

A common use of these models is where we have repeated measures data and the measurement occasions are close together in time.



The above graph shows a linear time trend fitted to repeated measurements on one individual. We can see that the residuals around the graph are not independent. Residuals close together in time show positive correlations; this correlation decreases as the time distance between measurements increases. In a multilevel analysis we will have many such lines, one for each individual in the dataset. In the multilevel case too, the residuals around each person's line may show a pattern of non-independence which is a violation of our model assumptions and could potentially lead to incorrect estimates of parameters. The covariance between two measurements taken at occasions i_1 and i_2 on individual j cannot be assumed to be 0. That is

$$\mathrm{cov}(e_{i_1 j}, e_{i_2 j}) \neq 0$$

We expect this covariance to decrease as the time interval between the measurements increases. Let t_{ij} denote the time of the ith measurement on the jth individual. A natural model for the covariance is

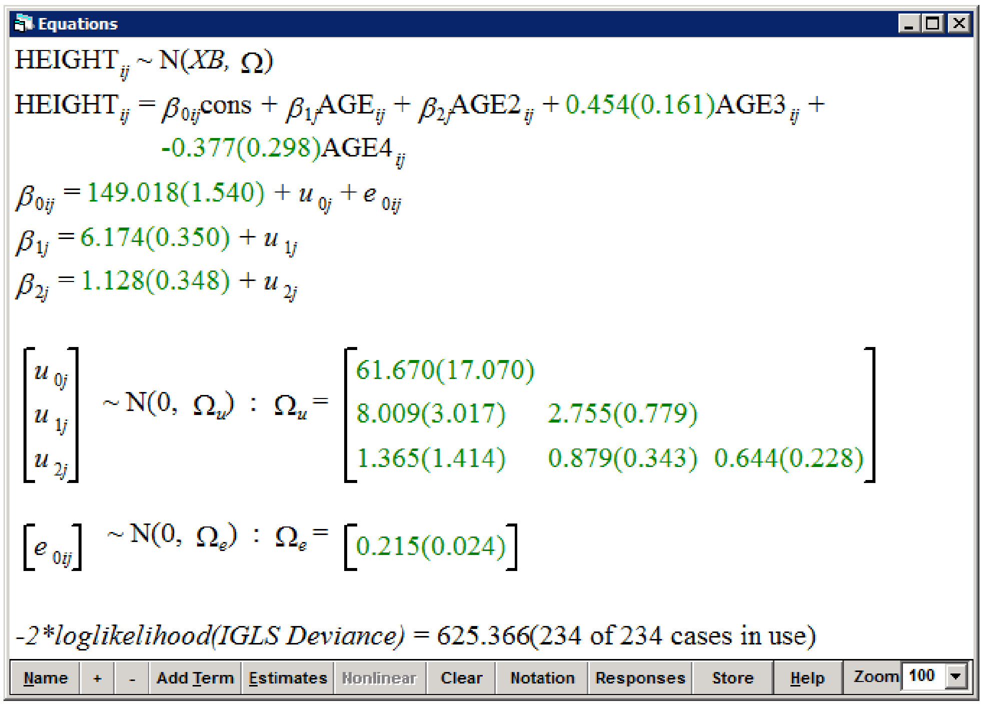
$$\mathrm{cov}(e_{i_1 j}, e_{i_2 j}) = \alpha \frac{1}{|t_{i_1 j} - t_{i_2 j}|}$$

The autocorrelation is then

$$\mathrm{cor}(e_{i_1 j}, e_{i_2 j}) = \frac{\alpha \frac{1}{|t_{i_1 j} - t_{i_2 j}|}}{\sigma_e^2} \qquad (1)$$

We will use the Oxford boys dataset to illustrate how to fit multilevel time series models with autocorrelated errors modelled as (1). The Oxford boys dataset contains height measurements on 26 boys each measured on nine occasions between the ages of 11 and 13.

First we set up a repeated measures model on the **oxboys** data with no autocorrelation structure. The data is in the worksheet **oxboys.ws**. The following model should already be set up in the **Equations** window:



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

$$\text{HEIGHT}_{ij} = \beta_{0ij}\text{cons} + \beta_{1j}\text{AGE}_{ij} + \beta_{2j}\text{AGE2}_{ij} + 0.454(0.161)\text{AGE3}_{ij} + -0.377(0.298)\text{AGE4}_{ij}$$

$$\beta_{0ij} = 149.018(1.540) + u_{0j} + e_{0ij}$$

$$\beta_{1j} = 6.174(0.350) + u_{1j}$$

$$\beta_{2j} = 1.128(0.348) + u_{2j}$$

$$\begin{bmatrix} u_{0j} \\ u_{1j} \\ u_{2j} \end{bmatrix} \sim \text{N}(0, \Omega_u) : \Omega_u = \begin{bmatrix} 61.670(17.070) & & \\ 8.009(3.017) & 2.755(0.779) & \\ 1.365(1.414) & 0.879(0.343) & 0.644(0.228) \end{bmatrix}$$

$$\begin{bmatrix} e_{0ij} \end{bmatrix} \sim \text{N}(0, \Omega_e) : \Omega_e = \begin{bmatrix} 0.215(0.024) \end{bmatrix}$$

*-2*loglikelihood(IGLS Deviance)* = 625.366(234 of 234 cases in use)

To add the term (1) to the model we first construct

$$\delta_{(i_1,i_2)j} = |t_{i_1 j} - t_{i_2 j}|$$

In this dataset we have 26 individuals each with 9 measurements so δ is a list of 26 symmetric matrices of dimension 9x9. That is

$$\begin{bmatrix} t_{1,1} - t_{1,1} & & & \\ t_{2,1} - t_{1,1} & 0 & & \\ \vdots & \vdots & \ddots & \\ t_{9,1} - t_{1,1} & t_{9,1} - t_{2,1} & \cdots & t_{9,1} - t_{9,1} \end{bmatrix}$$

$$\vdots$$

$$\begin{bmatrix} t_{1,26} - t_{1,26} & & & \\ t_{2,26} - t_{1,26} & 0 & & \\ \vdots & \vdots & \ddots & \\ t_{9,26} - t_{1,26} & t_{9,26} - t_{2,26} & \cdots & t_{9,26} - t_{9,26} \end{bmatrix}$$

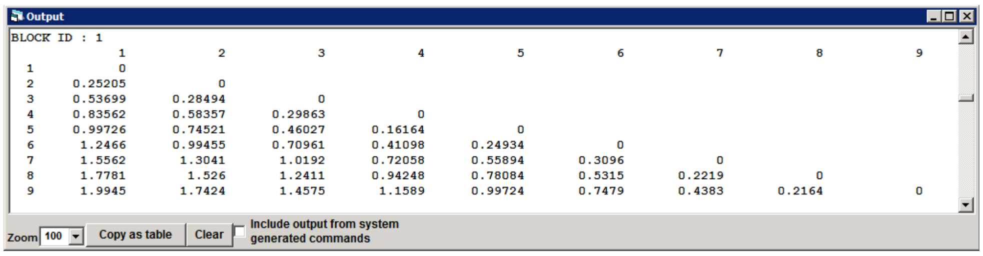
The SUBS command can set up such difference matrices (see ‘command SUBS’ in MLwiN Help for details on how to use this command). In the **Command interface** window type

```
▶ subs c1 -1 c2 c2 c10
```

In this format the command calculates the required 26 symmetric matrices and stacks them in c10. These matrices can be viewed with the MVIEw command:

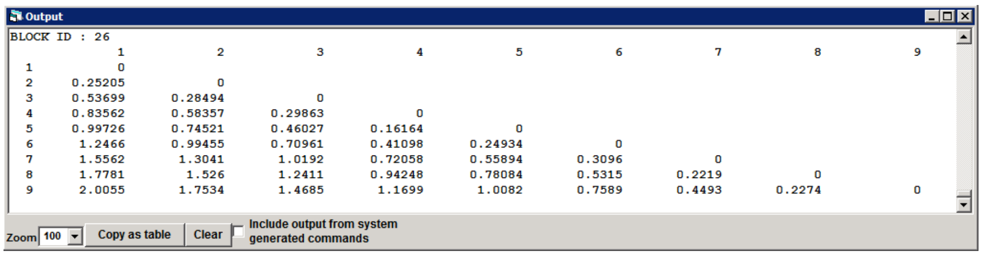
```
▶ MVIE 'ID' c10
```

which produces, after clicking the **Output** button in the Command interface window:



```
BLOCK ID : 1
               1           2           3           4           5           6           7           8           9
  1            0
  2      0.25205           0
  3      0.53699     0.28494           0
  4      0.83562     0.58357     0.29863           0
  5      0.99726     0.74521     0.46027     0.16164           0
  6       1.2466     0.99455     0.70961     0.41098     0.24934           0
  7       1.5562      1.3041      1.0192     0.72058     0.55894      0.3096           0
  8       1.7781       1.526      1.2411     0.94248     0.78084      0.5315      0.2219           0
  9       1.9945      1.7424      1.4575      1.1589     0.99724      0.7479      0.4383      0.2164           0
```

⋮



```
BLOCK ID : 26
               1           2           3           4           5           6           7           8           9
  1            0
  2      0.25205           0
  3      0.53699     0.28494           0
  4      0.83562     0.58357     0.29863           0
  5      0.99726     0.74521     0.46027     0.16164           0
  6       1.2466     0.99455     0.70961     0.41098     0.24934           0
  7       1.5562      1.3041      1.0192     0.72058     0.55894      0.3096           0
  8       1.7781       1.526      1.2411     0.94248     0.78084      0.5315      0.2219           0
  9       2.0055      1.7534      1.4685      1.1699      1.0082      0.7589      0.4493      0.2274           0
```

Now we form $\frac{1}{|t_{i_1j}t_{i_2j}|}$

Command	**Explanation**
`chan 0 c10 -1 c10`	avoid zero divide
`calc c11=1/c10`	
`chan -1 c11 0 c11`	set diagonal back to 0

C10 now contains the required structure to be added to the covariance matrix automatically specified by the multilevel model

$$\text{cov}(e_{i_1j}, e_{i_2j}) = \alpha \frac{1}{|t_{i_1j} - t_{i_2j}|}$$

where $\frac{1}{|t_{i_1j}t_{i_2j}|}$ are known and stored in c11, and α is to be estimated. The command

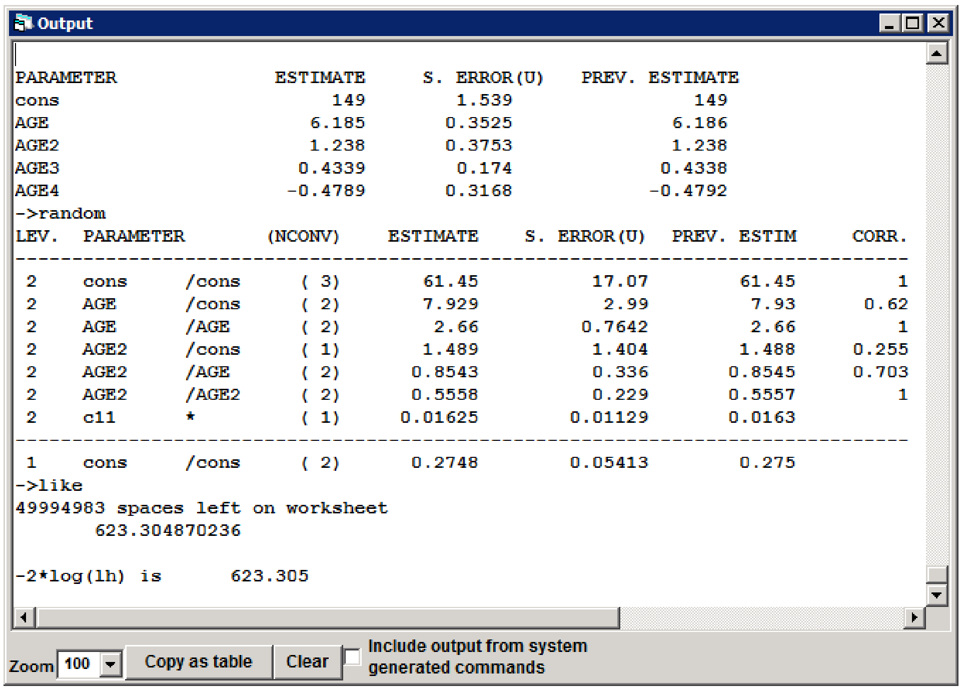
```
▶ SETDesign 2 C11
```

adds the design matrix held in C11 to the model at level 2. It may seem odd that this design matrix is applied at level 2 even though we are modelling autocorrelation between level 1 errors. This is because MLwiN thinks of any design matrices modelling non-independence between level 1 units as a higher level phenomenon.

At the moment the Equations window does not show the matrices specified via SETD and their associated parameter estimates. We have to revert to the Command interface to see them. In the **Command interface** window type

Command	Explanation
`batc 1`	don't pause in between iterations
`maxi 50`	set maximum no. of iterations to 50
`star`	run model
`fixed`	print out fixed parameters
`random`	print out random parameters
`like`	print out log likelihood

which updates the text **Output** window:



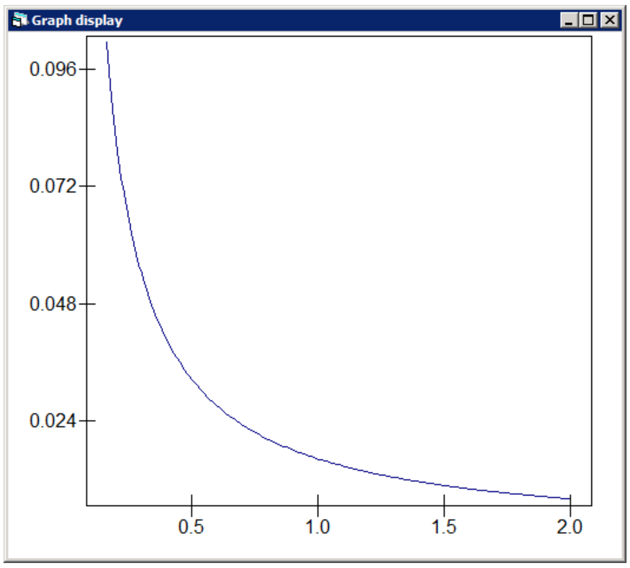
```
PARAMETER                    ESTIMATE      S. ERROR(U)    PREV. ESTIMATE
cons                              149          1.539               149
AGE                             6.185         0.3525             6.186
AGE2                            1.238         0.3753             1.238
AGE3                           0.4339          0.174            0.4338
AGE4                          -0.4789         0.3168           -0.4792
->random
LEV.   PARAMETER              (NCONV)     ESTIMATE    S. ERROR(U)   PREV. ESTIM      CORR.
--------------------------------------------------------------------------------
 2     cons      /cons        ( 3)         61.45          17.07          61.45          1
 2     AGE       /cons        ( 2)         7.929           2.99           7.93       0.62
 2     AGE       /AGE         ( 2)          2.66         0.7642           2.66          1
 2     AGE2      /cons        ( 1)         1.489          1.404          1.488      0.255
 2     AGE2      /AGE         ( 2)        0.8543          0.336         0.8545      0.703
 2     AGE2      /AGE2        ( 2)        0.5558          0.229         0.5557          1
 2     c11       *            ( 1)       0.01625        0.01129         0.0163
--------------------------------------------------------------------------------
 1     cons      /cons        ( 2)        0.2748        0.05413          0.275
->like
49994983 spaces left on worksheet
       623.304870236

-2*log(lh) is      623.305
```

The estimate of α is 0.01625. The range of $t_{i_1j} - t_{i_2j}$ is from 0.16 to 2 years. So we can generate the autocorrelation function by

Command	Explanation
`gene 0.16 2 0.01 c100`	generate $t_{i_1j} - t_{i_2j}$
`calc c101 = 0.01625*(1/c100)`	calculate $\alpha^{\frac{1}{\lvert t_{i_1j} - t_{i_2j}\rvert}}$
`calc c101 = c101/0.275`	

Then plotting c101 against c100 gives:



The drop in the likelihood from adding the autocorrelation parameter to this model is only 2. So in this case the extra term is not required.

6 Saving and retrieving of Minitab, Stata and SPSS work files

MLwiN now provides the following additional data file types on the Save / Open worksheet dialogue boxes:

Stata (*.dta files, versions 5 - 12)
SPSS (*.sav files, up to version 14)
Minitab (*.mtw, versions 12 and 13)
SAS transport (*.xpt)

Data, missing data values, variable names and category names are transferred.

Commands:

Command	Function
RSTAta filename	-open a Stata file
SSTAta filename	-save as a Stata file
RSPSs filename	-open an SPSS file
SSPSs filename	-save as an SPSS file
RMTW filename	-open a Minitab file
SMTW filename	-save as a Minitab file
RSAS filename	-open a SAS transport file
SSAS filename	-save as a SAS transport file

7 Zipped MLwiN worksheets

MLwiN can now save and open zipped versions of MLwiN worksheets (*.wsz). This format can reduce disc space usage by between 95% and 99%. Saving as a zipped version is the default option when selecting Save from the File menu; there are also commands which will save and open zipped versions of worksheets.

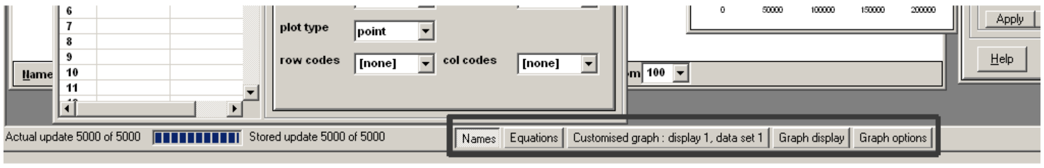
Commands:

Command	Function
ZRETr filename	-open a zipped worksheet
ZSAVe filename	-save worksheet in zipped form

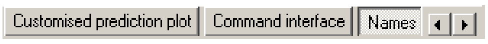
8 Other new features

8.1 Window tabs

There is now a series of tabs shown at the bottom of MLwiN, with one for each window open. Clicking on these provides an easy way to bring up the required window.

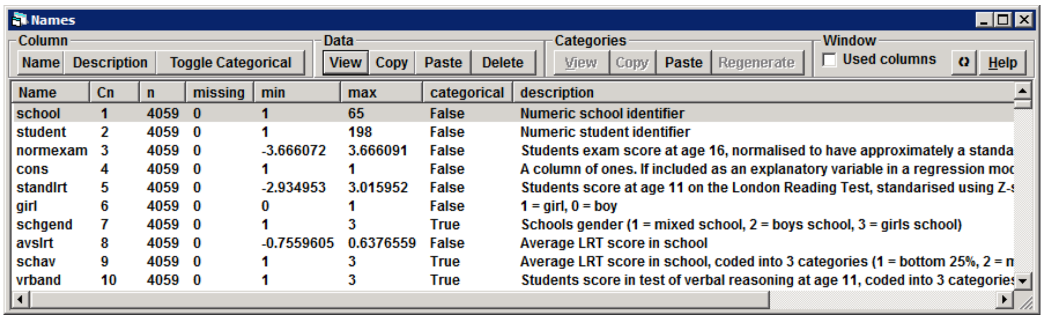


When there are too many windows open for all the tabs to be able to be shown at once, a pair of arrow buttons are shown which allow the user to scroll through the tabs:



8.2 The Names window

The new Names window has some extra features for ease of usage. The new Names window looks like this:



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

8.2.1 Column buttons

The **Name** button under **Column** allows the name of the highlighted column to be changed.

You can now add descriptions for each variable in the worksheet by selecting a column in the **Names** window and pressing the **Description** button.

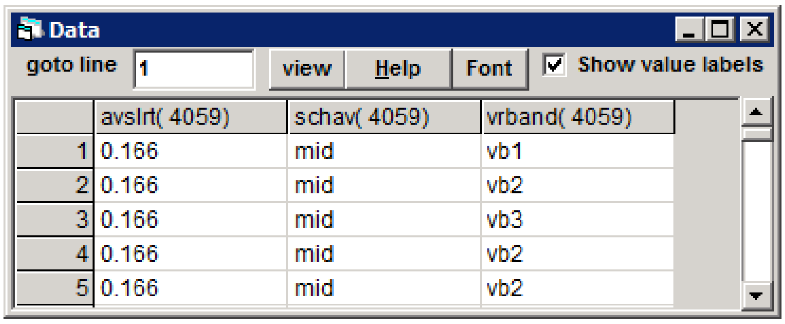
See section 8.2.3 for documentation of the **Toggle Categorical** button.

8.2.2 Data buttons

You can now highlight variables in the Names window and view the corresponding data directly by pressing the **View** button under **Data**. For example (using the **tutorial** dataset),

- Highlight columns 8–10 (**avslrt**, **schav**, **vrband**) in the **Names** window (use ctrl + click to select multiple columns)
- Click **View** under **Data**

This produces:



	avslrt(4059)	schav(4059)	vrband(4059)
1	0.166	mid	vb1
2	0.166	mid	vb2
3	0.166	mid	vb3
4	0.166	mid	vb2
5	0.166	mid	vb2

The **Copy** and **Paste** buttons under **Data** can be used to copy and paste variables from column to column in the worksheet, from one MLwiN worksheet to another, or to/from other applications such as Excel spreadsheets. To copy, highlight a column or columns (which need not be consecutive) and click **Copy** under **Data**. The column name is also copied and placed into the first row of the relevant column (the data the column contains being shifted down to row 2 onwards). To paste, first the destination columns must be highlighted. You can highlight the required number of columns, which need not be consecutive; in this case any data already present in these columns will be overwritten. Alternatively, you can highlight just one column; in this case the first column of data will be pasted into this column, whether it contains data or not, and the remaining columns will be pasted into the next free columns in the worksheet after this one. After highlighting, click **Paste** under **Data**. See section 8.2.3 for more details on copying and pasting categorical variables. If the first row of the pasted data contains a non-numerical string, this will be used as the column name; otherwise the column keeps the name it already has. This means that if copying and pasting from column to column within a worksheet, you will get an error message when you paste warning you that a column with the name you are trying to assign to the column you are pasting into already exists "duplicate name(PAST)". Simply click ok, and the column will retain the name it had before you pasted into it. You can then change this name if you wish to something else (that is not already used as a name by another column in the worksheet).

The **Delete** button will erase all the data in the highlighted column(s) and rename them as c<column number>.

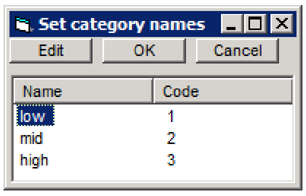
8.2.3 Categorical variables

There is now a column in the Names window that indicates whether a variable is categorical or not. The **Toggle Categorical** button will toggle the categorical status of a variable. When converting a non-categorical variable to categorical using this button, category names are automatically generated. These will be of the form <variable name>_<category number>, e.g. **ethnicity_1**. The category labels can be viewed and edited using the **View** under **Categories**, as we will see shortly. (Note that from v2.10 to v2.17, this function was performed by the **Categories** button).

Beware unexpected behaviour for categorical variables. When the underlying data in a column defined as categorical changes MLwiN does not update the list of category names associated with that column. To illustrate this:

- Select the **schav** variable in the **Names** window
- Click the **View** button in the categories section

This shows **schav** is a 3 category variable



Name	Code
low	1
mid	2
high	3

- Click the **OK** button

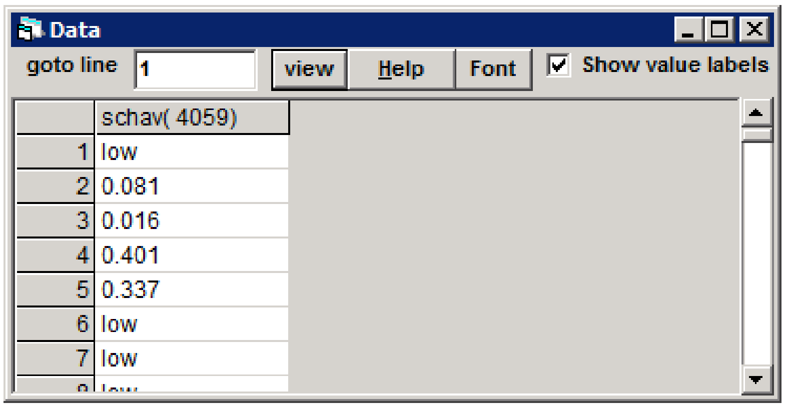
Let's now overwrite the data in this column with a set of uniform random numbers:

- In the **Command interface** window type:

```
▶ uran 4059 'schav'
```

- Now click the **Data** button in the **Names** window

This shows the following odd mixture of numbers and category names for column 9



	schav(4059)
1	low
2	0.081
3	0.016
4	0.401
5	0.337
6	low
7	low

What has happened is that the category information for column 9 has persisted and where any numbers in column 9, rounded to the nearest integer, correspond to a category number the name for that category is displayed. You can verify this by deselecting the **Show value labels** tick box in the **Data** window.

Whenever the underlying data for a categorical variable changes in such a way that the categories present in the data change, you need to update the category information. Changes to the underlying data could happen through a recoding (such as we just carried out), or when selecting or omitting certain cases, for example omitting all cases belonging to a certain category. Updating the category information would mean toggling the categorical status variable off if the variable has become continuous. If instead the variable remains categorical, you can **Toggle categorical** off, **Toggle categorical** on and then, if non-default category names are required, use **View** under **Categories** to re-enter the category names.

A shortcut when some of the categories remain the same is the **Regenerate** button. This retains category names for numbers which are still present in that variable, discards the category names for any numbers which are no longer present in the variable, and adds automatically generated category names for any numbers which did not already have category names assigned but are now present in the variable. These automatically generated names (if any) can then if desired be edited as described above using the **View** button under **Categories**. Take care when using the **Regenerate** button after recoding the underlying numerical values to be sure that any retained category names apply to the appropriate category numbers (since the numbers the names should apply to may change when you recode).

The **Regenerate** button may also be useful after importing data to MLwiN from other programs such as SPSS or Stata which allow several different categories of MISSING. On importing, MLwiN will recode all these values to its MISSING value but the labels remain assigned to their original codes, causing extra unwanted dummies or equations to appear when entering the variable into a model as an explanatory or response variable. Highlighting each variable in turn and pressing the **Regenerate** button will remove the unneeded category labels.

Category labels can be copied from a variable by highlighting this variable and pressing **Copy** under **Categories**. These can then be pasted onto another column by highlighting that column and pressing **Paste** under **Categories**. The column pasted to need not already be categorical; if it is not then it will become categorical when the categories are pasted into it. (Note that there is no undo button for this paste operation). Category labels can also be pasted into a program such as Word Pad, Excel, or Word, edited here if desired, then copied from these programs and pasted back into MLwiN.

Copying a categorical variable (not just its labels) to another column in the same worksheet or to a different worksheet and preserving its category labels is a two step task. First the data must be copied, by highlighting the variable, pressing **Copy** under **Data**, highlighting the destination column, and pressing **Paste** under **Data**. This will copy the numerical values of the variable, but the variable will not be declared as categorical, and if **Toggle Categorical** is pressed it will not have the original category labels but instead automatically generated ones. To preserve the labels, the second step is, after pasting, to highlight the original variable again, press **Copy** under **Categories**, highlight the new variable, and press **Paste** under **Categories**.

8.2.4 New commands

We document here new commands in v2.1* and v2.2* which carry out the same functions as the buttons in the **Names** window.

`DESC` **C** 'description'

assigns a description to a column

e.g.

```
▶ DESC c10 'This variable was collected from...'
```

COPY Mode **N C..C**
N = 0/1 exclude/include column headings

copies listed variables into the clipboard with tab delimited format

`PASTe` **C..C**

pastes clipboard data into listed columns. If there are fewer columns provided in the command than columns of data on the clipboard then writing of data continues from the last column number supplied (note that in this case, the data is not only pasted into free columns: the writing of data continues into consecutive columns whether or not they already contain data).

`CCAT` **C**

copies the category labels from the specified column

`PCAT` **C**

pastes the category labels on the clipboard to the specified column

`RCAT` **C**

regenerates the category labels for the specified column

8.3 New data manipulation windows and commands

8.3.1 Combining categorical columns

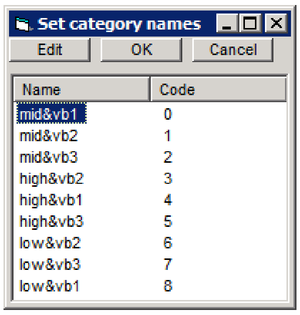
The COMBine command combines 2 or more columns containing categorical data. The categories of the output column will consist of a category for each possible different combination of the input codes with the names of these categories formed from the concatenation of the input category names. For example, given we have variables schav (1 = low, 2 = mid, 3 = high) and vrband (1 = vb1, 2 = vb2, 3 = vb3)

- If you carried out the demonstration in 8.1.2 showing how category names persist after changing the data in the **schav** column, then close the worksheet without saving and re-open it so you will have the correct data for **schav**
- In the **Command interface** window type:

```
▶ comb 'schav' 'vrband' c11
```

- Select C11 in the **Names** window
- Click the **View** button in the categories section of the **Names** window

This will produce:



Name	Code
mid&vb1	0
mid&vb2	1
mid&vb3	2
high&vb2	3
high&vb1	4
high&vb3	5
low&vb2	6
low&vb3	7
low&vb1	8

8.3.2 Finding unique codes

The UNIQue command will find each different value that occurs in the input column and place it just once in the output column (regardless of how many times it appears in the input column). For example, typing the command

```
▶ UNIQue 'normexam' c12
```

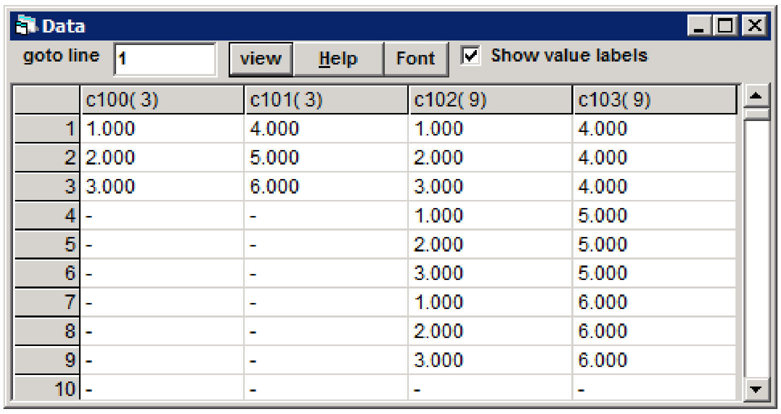
and then looking at the **Names** window shows that c12 contains 71 unique values in the range (−3.66, 3.66). This will have come about because our "continuous" variable Normexam was originally formed by applying a normal score transformation to a discrete scale with data on 71 points.

8.3.3 Creating numeric grids

It is sometimes useful to create a set of output columns containing all combinations of values occurring in a set of input columns. The UCOM command does this. For example,

```
▶ join 1 2 3 c100
▶ join 4 5 6 c101
▶ UCOM c100 c101 c102 c103
```

would create



	c100(3)	c101(3)	c102(9)	c103(9)
1	1.000	4.000	1.000	4.000
2	2.000	5.000	2.000	4.000
3	3.000	6.000	3.000	4.000
4	-	-	1.000	5.000
5	-	-	2.000	5.000
6	-	-	3.000	5.000
7	-	-	1.000	6.000
8	-	-	2.000	6.000
9	-	-	3.000	6.000
10	-	-	-	-

8.3.4 Recoding variables with short sequences of codes

The existing Recode window is convenient for discretising continuous variables according to a set of ranges. When we have variables with a small number of values an interface which lists each unique value and allows a new value to be specified is more helpful for recoding or merging short sequences of codes. Both these options are now available on the Recode sub-menu of the Data Manipulation menu.

Note that when you recode a categorical variable the category code information is not updated. So if you recode a variable so as to collapse 4 categories into 3, the variable will still be considered to have 4 categories (though one will have zero observations); or if you recode all observations in category 3 to have the value 10 and you do not already have a category with code 10, then category 3 will still have code 3 (and will have no observations) and observations with code 10 will not be considered to belong to any category. In order to update the category information after recoding you will need to highlight the variate in the **Names** window and press **Toggle Categorical** twice. This switches the variable to continuous and back to categorical, and when it is switched back to categorical the category names are re-created (if you have specific names you want to give the categories you will need to re-enter these by clicking on **Categories**).

8.3.5 Unvectorising data

For repeated measures analyis MLwiN requires the data to be structured one row per occasion, and for multivariate response modelling MLwiN automatically structures the data with one row per response variable. Sometimes it useful to take MLwiN data with one row per multivariate response or one row per occasion and *unvectorise* it, that is, turn it back so that it has one row per individual. This can be done with the UNVEctorise command:

`UNVEctorise` **N** stacked variables, stacked variable indicators in **C**, repeated individual codes in **C**, stacked data values in **C**, unique individual codes to **C**, unstacked data to **C**..**C**

Which is a bit of a mouthful. An example will help. Given the stacked data

Individual ID	Indicator	Value
C1	C2	C3
1	1	11
1	2	12
1	3	14
2	1	16
2	3	18

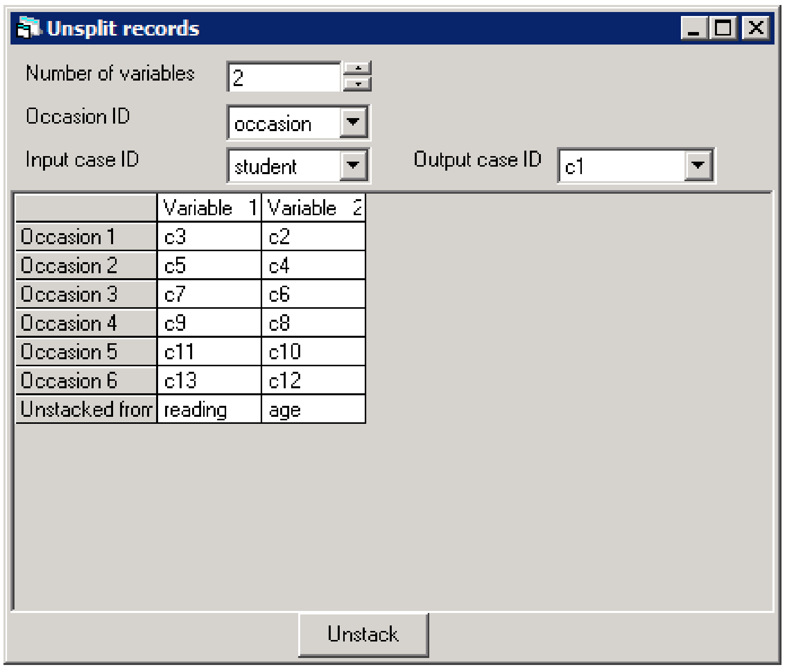
```
▶ UNVEct 3 c2 c1 c3 c4 c5 c6 c7
```

will produce

C4	C5	C6	C7
1	11	12	14
2	16	MISSING	18

The **Unsplit Records** menu item on the **Data Manipulation** menu provides a window to help specify the UNVEct command. It also makes it possible to unstack several variables at once, which cannot be done with the UNVEct command (to unstack multiple variables you have to use the command repeatedly). To demonstrate how to use the window:

- Open the **reading1.ws** worksheet
- Split the data as shown on pages 179–182 of the MLwiN User's Guide.
- Delete columns **c1–c13**
- Choose **Unsplit Records** from the **Data Manipulation** menu
- Set up the **Unsplit records** window as follows:



Number of variables	2
Occasion ID	occasion
Input case ID	student
Output case ID	c1

	Variable 1	Variable 2
Occasion 1	c3	c2
Occasion 2	c5	c4
Occasion 3	c7	c6
Occasion 4	c9	c8
Occasion 5	c11	c10
Occasion 6	c13	c12
Unstacked from	reading	age

- Click **Unstack**
- Select **No** when the box appears asking if you want to save the worksheet.
- Ignore the warning message that appears.

c1–c13 will now contain the data as it was when you opened the worksheet.

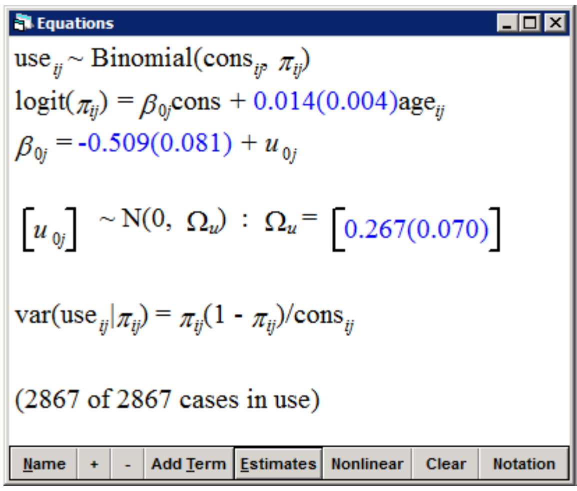
8.4 Macro Programming

8.4.1 Executing models from macros

You can run models from macros using the STARt and NEXT commands. By default the Equations window is not updated. Let's set up and run a couple of basic binary response models on the Bangladeshi fertility data from a macro. Open the worksheet bang.ws, then type or copy and paste the sequence of commands in the left column of the table below into an MLwiN macro

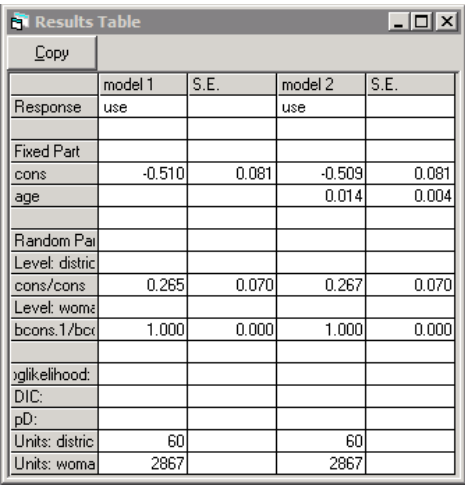
Command	Explanation
`mwipe`	clear model comparison table
`wset 15 1`	show the Equations window
`resp 'use'`	declare response
`rdist 1 0`	set distribution for first (and only) response to be binomial
`lfun 0`	set link function to logit
`linea 1 2`	set linearisation to 2nd order PQL
`doffs 1 'cons'`	set denominator to column of 1's
`iden 1 'woman' 2 'district'`	declare level ids
`addt 'cons'`	add intercept
`setv 2 'cons'`	declare random intercepts
`maxi 50`	set maximum iteration = 50
`batc 1`	don't pause in between iterations
`star`	run model
`msto 'model 1'`	
`addt 'age'`	add age
`next`	
`msto 'model 2'`	
`mcomp`	

If you execute this macro, the screen will not be updated until macro execution has been completed. You will then see the final model in the Equations window and both models in the model comparison table (click the **Estimates** button on the Equations window to see the results). (Note that the estimates in the Equations window always appear in blue when you run models from a macro, rather than in green, but this does not mean they have not converged).



$$\text{use}_{ij} \sim \text{Binomial}(\text{cons}_{ij}, \pi_{ij})$$
$$\text{logit}(\pi_{ij}) = \beta_{0j}\text{cons} + 0.014(0.004)\text{age}_{ij}$$
$$\beta_{0j} = -0.509(0.081) + u_{0j}$$
$$\left[u_{0j}\right] \sim \text{N}(0,\ \Omega_u) \ : \ \Omega_u = \left[0.267(0.070)\right]$$
$$\text{var}(\text{use}_{ij}|\pi_{ij}) = \pi_{ij}(1 - \pi_{ij})/\text{cons}_{ij}$$

(2867 of 2867 cases in use)



	model 1	S.E.	model 2	S.E.
Response	use		use	
Fixed Part				
cons	-0.510	0.081	-0.509	0.081
age			0.014	0.004
Random Pai				
Level: distric				
cons/cons	0.265	0.070	0.267	0.070
Level: woma				
bcons.1/bc	1.000	0.000	1.000	0.000
›glikelihood:				
DIC:				
pD:				
Units: distric	60		60	
Units: woma	2867		2867	

You may want to see the results after each model has been completed. In which case we place a PAUSe 1 command after the first STARt command in the macro. Even finer grained updating of the Equations (and other windows) is possible when running models from a macro. If you use the commands STARt and NEXT commands with the optional parameter 1, then windows are updated after each model estimation iteration.

It is sometimes useful to control the Equations window display from a macro. The following commands are useful for this

`NOTAtion` **N**: 0; 1 = Simple; General

`EXPAnd` **N**: 0; 1; 2; 3 = Show βs only; show βs and u_js; show βs, u_js and Ω_u; show βs, u_js, Ω_u and priors (if MCMC)

`NMVA` **N**: 0; 1 = display observed variables as symbols; display observed variables as names

`INDExing` **N**: 0; 1 = multiple subscript; single subscript

`ESTMates` **N**: 0; 1; 2 = symbols all black; symbols + convergence indication (blue, green); numbers + convergence indication

Other useful commands for specifying discrete response models in macros:

`RDISTribution` for response **N**, distribution type **M**
N is in range 1 to Number of responses, **M** = 0 binomial, 1 Poisson, 2 negative binomial, 3 Normal, 4 multinomial, 5 ordered multinomial

`LFUN` **N**: 0, 1, 2 = logit, probit, cloglog, log
If response distribution is Normal LFUN setting is ignored and identity link is used.

`LINEarise` **N M**: **N** is 0, 1 = MQL, PQL; **M** is 1, 2 = order 1, order 2 Taylor expansion

`DOFFs` for response **N** is in **C**: set denominator or offset column to **C**
N is in range 1 to Number of responses. If the response variable has a binomial or multinomial distribution the column is taken to contain denominators. If the response variable has a Poisson or Negative Binomial distribution, the column is taken to be the natural log of the required offset value.

8.4.2 Other New Commands

The following commands have been added / extended, which may be useful when programming in macros.

`SJOIn` **S** ‘text’ **B N**. . . **S** : string concatenation. For example:

```
► sjoin 'There are' s1
► set b1 7
► sjoin s1 ' ' b1 ' deadly sins' s2
► say s2
```

produces (in the **Output** window): `There are 7 deadly sins`

`NMSTr` **C S** : create a string consisting of the name of a column

for example:

```
► NAME c1 'hello'
► NMSTR c1 s1
► SAY s1
```

The following commands have been updated to work with string variables:

All worksheet save and retrieve commands
`PREf`
`POSTf`
`DESCription`
`SAY`
`WMSG`
`LOGOn`

```
MSTOre
MPRInt
MDELete
```

For example,

```
► SAVE c:\test.ws
```

is equivalent to

```
► SJOIN 'c:\test.ws' s1
► SAVE s1
```

8.5 Invoking MLwiN from a command line (or other packages)

MLwiN can be invoked from a command line with a number of potentially useful switches. Probably the main purpose of this is if MLwiN is to be called from an external scripting engine.

```
mlwin <switches> <worksheet name>

Switches:
/help  show list of switch options
/run  an mlwin macro file
/load  an mlwin macro into mlwin's macro file editor
/sheetsize  worksheet size in k cells
/levels  number of levels to allow in analysis
/columns  number of worksheet columns
/min  start MLwiN minimized
/nogui  hide MLwiN graphical interface
/reset  reset MLwiN settings to defaults
/nowarn  suppress warning messages
```

A Algorithm used for producing predicted means and intervals

Simulation techniques, for predicting means, intervals and coverage (from higher level variances), are used for all models even when there is an analytical solution (eg for Normal response models and cluster specific predictions from discrete repsonse models). This is not always computationally most efficient, but allows the same code for all model types.

A.1 Normal response models

$$\begin{aligned} y_{ij} &= (X\beta)_{ij} + Z_{ij}u_j^T + e_{ij} \\ u_j &\sim \mathrm{N}(0, \Omega_u) \\ e_{ij} &\sim \mathrm{N}(0, \sigma_e^2) \end{aligned}$$

Suppose the user wants to make predictions for a user specified set of values of explanatory variables. We have $x^{(c)}$, the set of user specified explanatory variable values for the cth prediction case and $z^{(c)}$, the set of user defined explanatory variables with random coefficients at level 2 (note that the code generalises to any number of hierarchical levels).

For case c we pick

$$\mathrm{B}_k \sim \mathrm{N}(\hat{\beta}, \mathrm{cov}(\hat{\beta})), \qquad k = 1, \ldots, K$$

and form

$$\hat{y}_{ck} = x_c \mathrm{B}_k$$

Thus for case c we have K predictions from which we derive required statistics eg mean, upper and lower confidence intervals eg

$$\hat{y}_c = \frac{1}{K} \sum_{k=1}^{K} \hat{y}_{ck}$$

If coverage intervals, from higher level variances, are required we pick M values of u from $\mathrm{N}(0, \hat{\Omega}_u)$. We then construct

$$\hat{y}_{cm} = x_c \hat{\beta} + Z_c u_m^T$$

from which we can derive coverage statistics for case c

A.1.1 Predictions and confidence intervals for differences

We often want predictions and confidence intervals not for means but for differences. For example given a model:

$$\begin{aligned} y_{ij} &= \beta_0 x_0 + \beta_1 x_{1ij} + \beta_2 x_{2ij} + \beta_3 x_{1ij} x_{2ij} + u_j + e_{ij} \\ u_j &\sim \mathrm{N}(0, \sigma^2_u) \\ e_{ij} &\sim \mathrm{N}(0, \sigma^2_e) \end{aligned}$$

we may want the difference curve with confidence intervals for $x_{2ij} = a$ and $x_{2ij} = b$, i.e. the curve giving the difference in y_{ij} for two different values of x_{2ij}, a and b, across a range of values of x_{1ij}. In the Normal case this can be derived analytically: the difference curve is given by the formula $\beta_2 x_{2\mathrm{d}ij} + \beta_3 x_{1ij} x_{2\mathrm{d}ij}$, where $x_{2\mathrm{d}ij} = a - b$; however the software uses simulation. For example if the user asks for the difference curve for $x_{2ij}=$ 0 and $x_{2ij}=$ 1 for $x_{1ij} = \{-3, 0, 3\}$, by asking for predictions for values $\{x_{1ij} = \{-3, 0, 3\}$; $x_{2ij} = \{0, 1\}\}$ and setting differences from $x_{2ij} = 0$, the software follows this procedure:

1. Simulate $\mathrm{B}_k \sim \mathrm{N}(\hat{\beta}, \mathrm{cov}(\hat{\beta})), \qquad k = 1 : K$
2. For each k, calculate $x_c\mathrm{B}_k$ for each combination x_c of values of the other explanatory variables (in this case, just x_{1ij}), for both $x_{2ij} = 0$ (call this $\hat{y}_{0ck}$) and $x_{2ij} = 1$ (call this $\hat{y}_{1ck}$). So in this example, for each k calculate

c	x_{1ij}	$\widehat{y_{0ck}}$	$\widehat{y_{1ck}}$
1	-3	$\beta_{k0} - 3\beta_{k1}$	$\beta_{k0} - e\beta_{k1} + \beta_{k2} - 3\beta_{k3}$
2	0	β_{k0}	$\beta_{k0} + \beta_{k2}$
3	3	$\beta_{k0} + 3\beta_{k1}$	$\beta_{k0} + 3\beta_{k1} + \beta_{k2} + 3\beta_{k3}$

3. Now for each c calculate

$$d_{ck} = \hat{y}_{0ck} - \hat{y}_{1ck}$$

and then

$$d_c = \frac{1}{K} \sum_{k=1}^{K} d_{ck}$$

is the difference for prediction case c, so the set $\{d_c\}$ is the set of values requested by the user. The software uses the same simulation process to calculate confidence intervals for differences, finding for each c the range of values between which d_{ck} lies for 95% of the B_k.

If coverage intervals are required, then in step 1 the software simulates M values of u from $\mathrm{N}(0, \hat{\Omega}_u)$. In step 2 the software calculates $x_c\hat{\beta} + Z_c u^T_m$ for each c, for $x_{2ij} = a$ (call this $\hat{y}_{acm}$) and $x_{2ij} = b$ (call this $\hat{y}_{bcm}$), and then the difference between these, $\hat{d}_{cm} = \hat{y}_{acm} - \hat{y}_{bcm}$.

$\hat{d}_c = \frac{1}{M} \sum_{m=1}^{M} \hat{d}_{cm}$ now gives the required coverage intervals.

A.2 Binomial models

As an example let's take a 2 level binomial model

$$\begin{aligned} y_{ij} &\sim \text{Binomial}(n_{ij}, \pi_{ij}) \\ f(\pi_{ij}) &= (X\beta)_{ij} + Z_{ij}u_j^T \\ u_j &\sim \text{N}(0, \Omega_u) \end{aligned}$$

Now we can get predictions on the raw (probabilities) or transformed (logit) scales and predictions can be for the median or mean

A.2.1 Median (Cluster specific)

As with Normal models we form

$$\text{B}_k \sim \text{N}(\hat{\beta}, \text{cov}(\hat{\beta})), \qquad k = 1 : K$$

Now for each k and each c, the median (cluster specific) predicted probability is

$$\hat{p}_{ck} = \text{antilogit}(x_c \text{B}_k)$$

and the median predicted probability from the cluster specific predictions is

$$\hat{p}_c = \text{median}(\hat{p}_{ck}) \text{ across } k$$

Again percentiles are derived from the set $\hat{p}_{ck}$.

Predictions and confidence intervals on the logit scale we derive from

$$\hat{y}_{ck} = x_c \text{B}_k$$

Required summary statistics for differences are calculated from the following:

$$\begin{aligned} \hat{d}_{ck}^{(y)} &= x_c \text{B}_k - (x_{c_r} \text{B}_k) \\ \hat{d}_{ck}^{(p)} &= \text{antilogit}\,(x_c \text{B}_k) - \text{antilogit}\,(x_{c_r} \text{B}_k) \end{aligned}$$

For coverage intervals from higher level variances, we pick M values of u from $\text{N}(0, \Omega_u)$. We then construct

$$\hat{p}_{cm} = \text{antilogit}(x_c \hat{\beta} + z_c u_m^T)$$

and required coverage intervals can be calculated from the $\hat{p}_{cm}$ chain and likewise for logit predictions. Coverage intervals for differences:

$$\hat{d}^{(y)}_{cm} = x_c\hat{\beta} + z_c u^T_m - (x_{c_r}\hat{\beta} + z_{c_r} u^T_m)$$
$$\hat{d}^{(p)}_{c,m} = \text{antilogit}(x_c\hat{\beta} + z_c u^T_m) - \text{antilogit}(x_{c_r}\hat{\beta} + z_{c_r} u^T_m)$$

A.2.2 Mean (Population average)

We pick M values of u from $\text{N}(0, \Omega_u)$; these M values are applied to each combination of x_c and B_k. So now

$$\hat{p}_{ck} = \frac{1}{M}\sum_m \text{antilogit}(x_c \text{B}_k + z_c u^T_m)$$

$$\hat{p}_c = \frac{1}{K}\sum_k^{\hat{p}_{ck}}$$

Again upper and lower intervals can be calculated from the $\hat{p}_c$ chain. On the population average logit scale we work with:

$$\hat{y}_{ck} = \text{logit}\,(\hat{p}_{ck})$$

and

$$\hat{y}_c = \frac{1}{K}\sum_k \hat{y}_{ck}$$

Coverage intervals from level 2 variance do not apply in population average models

Differences on the probability scale are given by

$$\hat{d}^{(p)}_{ck} = \frac{1}{k}\sum_k (\hat{p}_{ck} - \hat{p}_{c_r k})$$

and on the logit scale are given by

$$\hat{d}^{(y)}_{ck} = \frac{1}{K}\sum_k (\text{logit}\,(p_{ck}) - \text{logit}\,(p_{c_r k}))$$

A.3 Unordered Multinomial

With A categories and the Ath category as the base

$$f(p^{(s)}_c) = \log(p^{(s)}_c / p^{(A)}_c)$$

$$p^{(s)}_c = f'(\beta, s, c) = \begin{cases} \frac{\exp((x_c\beta)^{(s)})}{1+\sum_{a=1}^{A-1}\exp((x_c\beta)^{(a)})} & a \neq A \\ 1 - \sum_{a=1}^{A} p^{(a)}_c & \text{otherwise} \end{cases}$$

where s and a both index the categories of the multinomial response

$$\mathrm{B}_k \sim \mathrm{N}(\hat{\beta}, \mathrm{cov}(\hat{\beta})), \qquad k = 1 : K$$

Now

$$\hat{p}_{ck}^{(s)} = f'(\mathrm{B}^{(k)}, s, c)$$

and the median predicted probability from the cluster specific predictions is

$$\hat{p}_{c}^{(s)} = \frac{1}{K} \sum_{k} \hat{p}_{ck}^{(s)}$$

The simulation method gives us confidence intervals for all categories including the reference category A.

For cluster specific logits:

$$\hat{y}_{ck}^{(s)} = \begin{cases} 0 & s = A \\ (x_c \beta_k)^{(s)} & \text{otherwise} \end{cases}$$

and

$$\hat{y}_{c}^{(s)} = \frac{1}{K} \sum_{k} \hat{y}_{ck}^{(s)}$$

Required summary statistics for differences are calculated from the following chains:

$$\hat{d}_{ck}^{(y)(s)} = \begin{cases} 0 & s = A \\ (x_c \mathrm{B}_k)^{(s)} - (x_{r_c} \mathrm{B}_k)^{(s)} & \text{otherwise} \end{cases}$$

$$\hat{d}_{ck}^{(p)(s)} = f'(\mathrm{B}^{(k)}, s, c) - f'(\mathrm{B}^{(k)}, s, c_r)$$

For coverage intervals from higher level variances, we pick M values of u from $\mathrm{N}(0, \Omega_2)$. We then construct

$$\hat{p}_{cm}^{(s)} = f'(\beta, s, c, m) = \begin{cases} 1 - \sum\limits_{a=1}^{A-1} p_c^{(a)} & s = A \\ \\ \frac{\exp((x_c\beta)^{(s)} + (z_c u_m^T)^{(s)})}{1 + \sum\limits_{a=1}^{A-1} \exp((x_c\beta)^{(a)} + (z_c u_m^T)^{(a)})} & \text{otherwise} \end{cases}$$

and required coverage intervals can be calculated from the $\hat{p}_{cm}^{(s)}$ chain and likewise for logit predictions. Coverage intervals for differences:

$$\hat{d}_{cm}^{(y)(s)} = \begin{cases} 0 & s = A \\ \Big((x_c\hat{\beta})^{(s)} + (z_c u_m^T)^{(s)}\Big) - \Big((x_{r_c}\hat{\beta})^{(s)} + (z_{r_c} u_m^T)^{(s)}\Big) & \text{otherwise} \end{cases}$$

$$\hat{d}_{cm}^{(p)(s)} = f'(\hat{\beta}, s, c, m) - f'(\hat{\beta}, s, c, m)$$

A.3.1 Population average

We pick M values of u from $\mathrm{N}(0, \Omega_u)$, these M values are applied to each combination of c and k. So now

$$\hat{p}_{ck}^{(s)} = \frac{1}{M} \sum_m f'(\beta_k, s, c, m)$$

$$\hat{p}_c^{(s)} = \frac{1}{K} \sum_k \hat{p}_{ck}^{(s)}$$

Again upper and lower intervals can be calculated from the $\hat{p}_c$ chain. On the population average logit scale we work with:

$$\hat{y}_{ck}^{(s)} = \log\left(\hat{p}_{ck}^{(s)} / \hat{p}_{ck}^{(A)}\right)$$

and

$$\hat{y}_c^{(s)} = \frac{1}{K} \sum_k \hat{y}_{ck}$$

Coverage intervals from level 2 variance do not apply in population average models

Differences on the probability scale are given by

$$\hat{d}_{ck}^{(p)(s)} = \frac{1}{K} \sum_k \left(\hat{p}_{ck}^{(s)} - \hat{p}_{c_r k}^{(s)}\right)$$

and on the logit scale are given by

$$\hat{d}_{ck}^{(y)(s)} = \frac{1}{K} \sum_k \left(\log\left(\hat{p}_{ck}^{(s)} / \hat{p}_{ck}^{(A)}\right) - \log\left(\hat{p}_{c_r k}^{(s)} / \hat{p}_{c_r k}^{(A)}\right)\right)$$

A.4 Ordered Multinomial

$o_c^{(s)}$ is the cumulative probability for category s in an ordered set of categories

$$\begin{aligned} o_c^{(s)} &= \sum_{a=1}^{s} p_c^{(a)} \qquad s = 1 : A - 1 \\ f(o_c^{(s)}) &= \mathrm{logit}(o_c^{(s)}) \\ p_c^{(s)} &= f'(\beta, s, c) = \begin{cases} 1 & a = A \\ \mathrm{antilogit}(x_c \beta) & \text{otherwise} \end{cases} \end{aligned}$$

Since we are working with predictions now on the logit scale, we can proceed with customised predictions as described for the binary, logistic case (as in section A.2)

References

[1] Evans, I S (1977) The Selection of class intervals *Transactions of the Institute of British Geographers*, New Series, Vol. 2(1), pp. 98-124

[2] Goldstein, H., Rasbash, J., Healy, M. (1994) Multilevel time series models with applications to repeated measures data *Statistics in Medicine* Vol 13 Issue 16 pp 1643-1655

[3] Hedeker, D. and Gibbons, R.D. (2006) Chapter 5: Mixed-effects polynomial regression models for continuous outcomes, *Longitudinal Data Analysis*, Wiley, New York

[4] Scripter, M W (1970) Nested-means map classes for statistical maps, *Annals of the Association of American Geographers*, Vol. 60 (2), pp 385-393.