

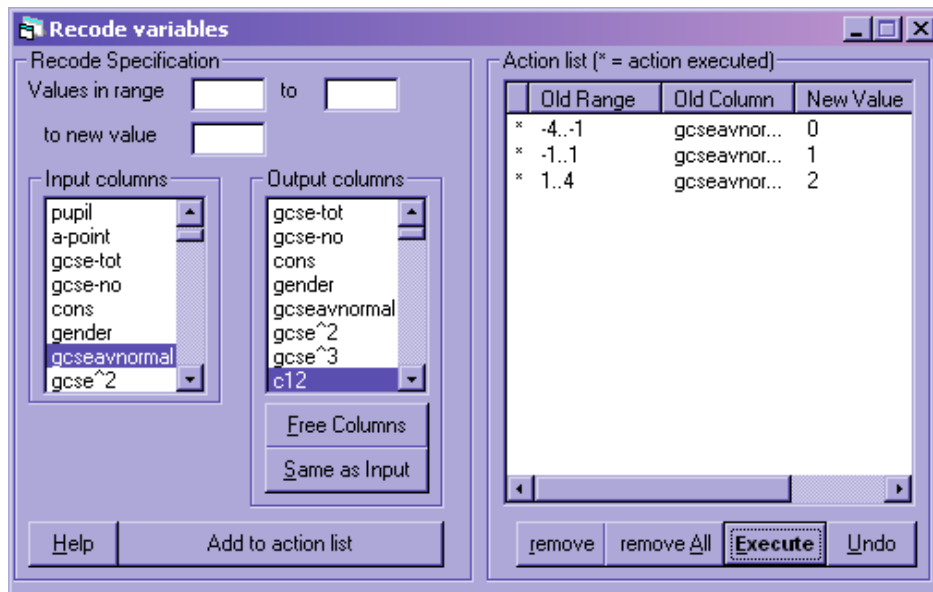
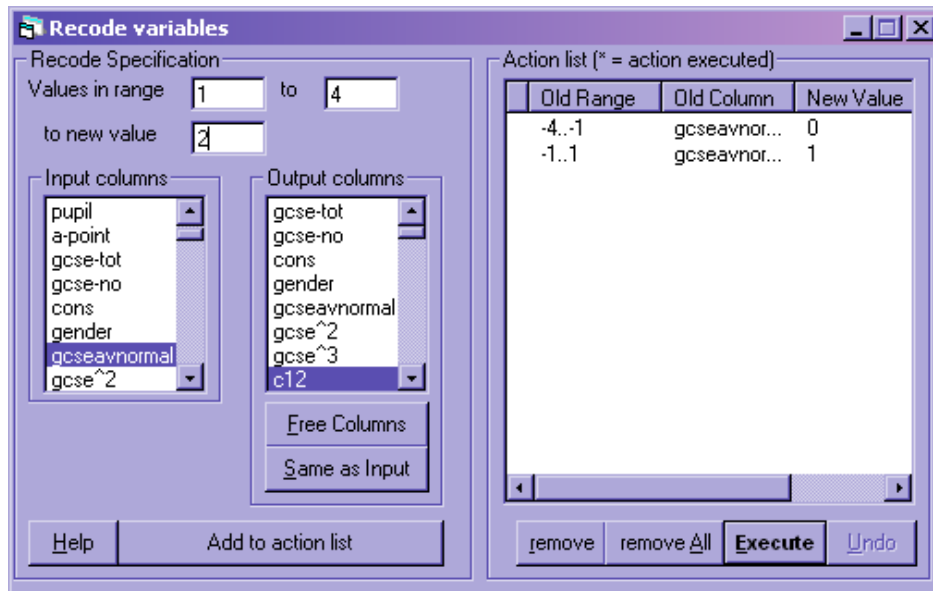
Using the Variance Function Window with Multinomial Models with Categorical Explanatory Variables

1 Ordered multinomial models

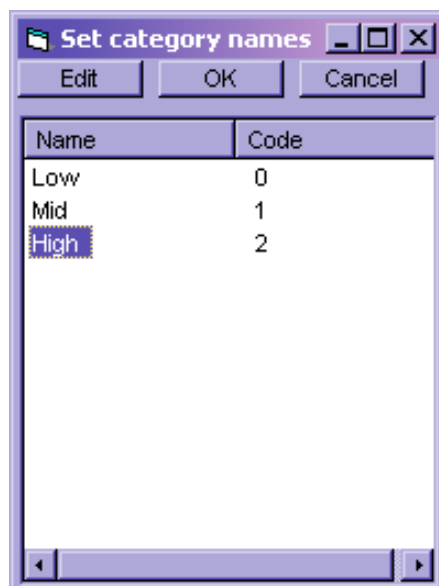
While ordered multinomial models have more difficult features for the learner to grasp, and are therefore usually presented after unordered multinomial models, when it comes to variance functions ordered multinomial models are actually simpler, since the variance function is the same for all response categories (unlike unordered multinomial models). Using the Variance function window for an ordered multinomial model is thus very like using it for a continuous response model; and so here we present ordered multinomial models first.

We will work with the **alevchem** worksheet that is included with MLwiN (you can access this by selecting **Open sample worksheet** from the **File** menu). We will create a categorical explanatory variable using the **gcseavnormal** variable which is generated on p164 of the User's Guide. Note that we do this so that we have a categorical explanatory variable for the purposes of the example; of course in real research it would not usually be sensible to convert a continuous variable to a categorical one in this way. We begin by creating this variable:

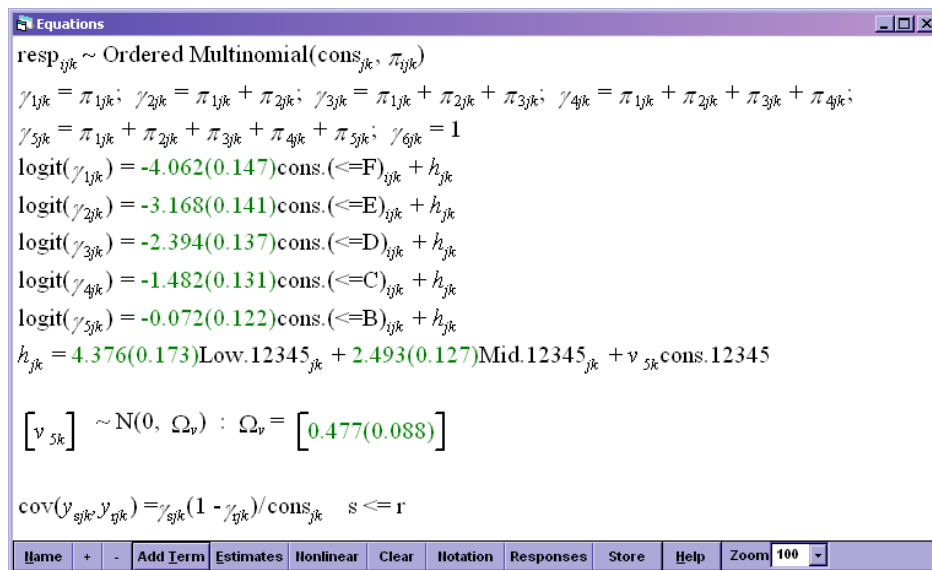
- From the **Data Manipulation** menu, select **recode** → **by Range**
- In the **Recode variables** window that appears, select **gcseavnormal** under **Input columns** and **c12** (or any free column) under **Output columns**
- In the **Values in range** box type -4 and in the **to** box type -1
- In the **to new value** box type 0
- Click **Add to action list**; this specifies that you want to recode all values in the range -4 to -1 appearing in **gcseavnormal** to new value 0, putting the recoded variable in c12
- Similarly, specify that values -1 to 1 should be recoded to 1 and values 1 to 4 should be recoded to 2
- Click **Execute**



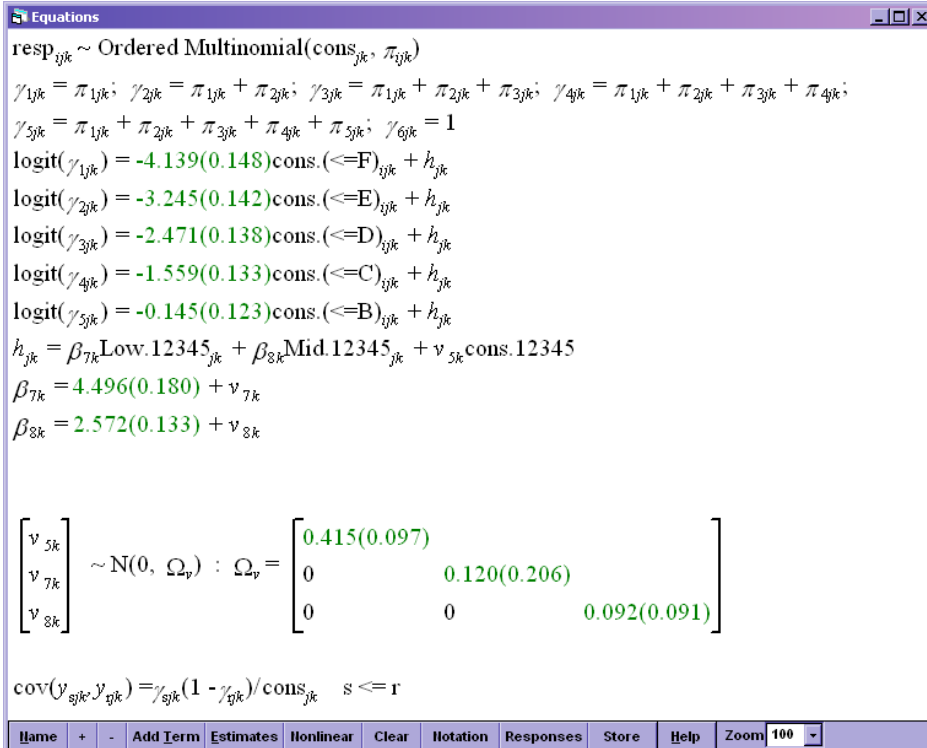
- In the **Names** window, rename **c12** to **gcse_cat**
- With this variable highlighted, click the the **Toggle Categorical** button
- Click on **View** under **Categories**
- Change the names of category 0 to **Low**, category 1 to **Mid**, and category 2 to **High**
- Click **OK**



- Set up and run a two-level ordered multinomial model, with response **a-point**, reference category **A**, and explanatory variables **cons** and **gcse_cat**, reference category **High**



- Put random slopes on the two dummy variables for **gcse_cat**
- Click on the off-diagonal terms in the establishment level variance-covariance matrix to set them to 0 (click **Yes** to the messages which appear).
- Run this model



Equations

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

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

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

$$\text{logit}(\gamma_{1jk}) = -4.139(0.148)\text{cons.}(\leq F)_{ijk} + h_{jk}$$

$$\text{logit}(\gamma_{2jk}) = -3.245(0.142)\text{cons.}(\leq E)_{ijk} + h_{jk}$$

$$\text{logit}(\gamma_{3jk}) = -2.471(0.138)\text{cons.}(\leq D)_{ijk} + h_{jk}$$

$$\text{logit}(\gamma_{4jk}) = -1.559(0.133)\text{cons.}(\leq C)_{ijk} + h_{jk}$$

$$\text{logit}(\gamma_{5jk}) = -0.145(0.123)\text{cons.}(\leq B)_{ijk} + h_{jk}$$

$$h_{jk} = \beta_{7k}\text{Low.12345}_{jk} + \beta_{8k}\text{Mid.12345}_{jk} + v_{5k}\text{cons.12345}$$

$$\beta_{7k} = 4.496(0.180) + v_{7k}$$

$$\beta_{8k} = 2.572(0.133) + v_{8k}$$

$$\begin{bmatrix} v_{5k} \\ v_{7k} \\ v_{8k} \end{bmatrix} \sim N(0, \Omega_v) : \Omega_v = \begin{bmatrix} 0.415(0.097) & & \\ 0 & 0.120(0.206) & \\ 0 & 0 & 0.092(0.091) \end{bmatrix}$$

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

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

The model converges after 76 iterations. The random slopes appear not to be significant judging by their standard errors (though to be sure we would have to use a Wald test or fit the model using MCMC and use the DIC), but since we are including them in order to demonstrate the use of the Variance function window with categorical explanatory variables and not for substantive reasons, we will leave them in. Note also that we have used IGLS and not MCMC to fit this model (indeed, we have used the least good IGLS estimation method, 1st order MQL); MCMC estimation is recommended for discrete response models (see this FAQ on the CMM website <http://www.cmm.bristol.ac.uk/MLwiN/tech-support/support-faqs/index.shtml#differentresults>), but IGLS can be used for model exploration, and we use it here for convenience because we are just demonstrating the use of the Variance function window (which is exactly the same whether using IGLS or MCMC) and not performing real research.

- From the Model menu, select **Variance function**
- From the **level** drop-down box, select **3:estab_long**

var($v_{5k}x_{5k} + v_{7k}x_{7jk} + v_{8k}x_{8jk}$) = $\sigma_{v5}^2 x_5^2 + \sigma_{v7}^2 x_{7jk}^2 + \sigma_{v8}^2 x_{8jk}^2$

select	cons.12345	Low.12345	Mid.12345	result

level: 3.estab_long | calc | **Name** | Help | Zoom: 100 | Copy

variance output to: [none] | 1.0 | SE of variance output to: [none]

The formula in the top pane of the window gives the establishment level variance. It is a function of **cons** and **gcseavnormal**. Currently, **cons** and **gcseavnormal** are represented by x_5 and x_6 ; to see the formula with variable names instead of x s, click the the **Name** button at the bottom of the window

var($v_{5k}cons.12345 + v_{7k}low.12345_{jk} + v_{8k}mid.12345_{jk}$) = $\sigma_{v5}^2 cons.12345^2 + \sigma_{v7}^2 Low.12345_{jk}^2 + \sigma_{v8}^2 Mid.12345_{jk}^2$

select	cons.12345	Low.12345	Mid.12345	result

level: 3.estab_long | calc | **Name** | Help | Zoom: 100 | Copy

variance output to: [none] | 1.0 | SE of variance output to: [none]

There are two facilities offered by the Variance function window. One is to calculate the variance for every case in the dataset, using the values of **cons** and of **gcseavnormal** for that case. To do this,

- From the **variance output to:** drop-down box select a free column
- Optionally, edit the 1.0 in the box in front of **SE of variance output to:** to the multiplier you want (e.g. 1.96 for 95% confidence intervals), and choose a free column from the **SE of variance**

output to: drop-down box; this column will then be filled with the multiplier times the standard error for each case, and can be used to plot error lines in a graph of the variance

- Click **calc**

If we now return to the **Names** window, we can see our new columns containing the variances and their standard errors. (It may be necessary to press the **refresh** button, which is next to the **Help** button at the top of the window, and has two arrows on it pointing round in a circle).

Name	Cn	n	missing	min	max	categorical	description
lea	1	2166	0	203	938	False	Local Education Authority (not...
estab	2	2166	0	4001	8603	False	Institution identification.
pupil	3	2166	0	1650	194909	False	Pupil identification.
a-point	4	2166	0	1	6	True	A level point score (see below...
gcse-tot	5	2166	0	22	92	False	Total point score for GCSE ex...
gcse-no	6	2166	0	5	12	False	Number of GCSE exams taken.
cons	7	2166	0	1	1	False	Constant (= 1)
gender	8	2166	0	0	1	True	1 if female, 0 if male.
gcseavnormal	9	2166	0	-3.502057	3.113836	False	
gcse^2	10	2166	0	5.357973E-04	12.26441	False	
gcse^3	11	2166	0	-42.95065	30.19167	False	
gcse_cat	12	2166	0	0	2	True	
resp	13	10830	0	0	1	False	
resp_indicator	14	10830	0	1	5	True	
bcons.1	15	10830	0	1	1	False	
pupil_long	16	10830	0	1650	194909	False	
estab_long	17	10830	0	4001	8603	False	
denom	18	10830	0	1	1	False	
cons.(≤F)	19	10830	0	0	1	False	
cons.(≤E)	20	10830	0	0	1	False	
cons.(≤D)	21	10830	0	0	1	False	
cons.(≤C)	22	10830	0	0	1	False	
cons.(≤B)	23	10830	0	0	1	False	
cons.12345	24	10830	0	1	1	False	
Low.12345	25	10830	0	0	1	False	
Mid.12345	26	10830	0	0	1	False	
c27	27	10830	0	0.4146065	0.5341755	False	
c28	28	10830	0	0.1894983	0.428362	False	
c29	29	0	0	0	0	False	
c30	30	0	0	0	0	False	
c31	31	0	0	0	0	False	
c32	32	0	0	0	0	False	
c33	33	0	0	0	0	False	

We can rename these columns to **variance** and **variance_se**. We can see that each column has a length of 10830. This is the length of the expanded dataset that MLwiN created to fit the multinomial model, and not the length of our original dataset (which was 2166). This is because the Variance function window when used in this way always produces one entry per lowest level unit of the model, and in this case the lowest level is not the conceptual lowest level, **pupil**, but rather **resp_indicator**. Our variance column thus has one entry per response category¹ per pupil. Since there are actually only three different values of the variance (one for **gcse_cat** = **Low**, one for **Mid**, and one for **High**), and we have 10830 entries, this is clearly rather inefficient in terms of storage. This is one reason why we might want to use the other facility offered by the Variance function window, which we will come to shortly.

We will plot the variances we have just calculated. Since the variances differ according to values of **gcse_cat**, we would like to plot them against this. However, we cannot plot them against **gcse_cat** itself, since **gcse_cat** has one entry per pupil and **variance** has one entry per response category per pupil. We therefore need to create a longer version of **gcse_cat** which also has one entry per response category per pupil. (Note that if our explanatory variable had been continuous instead of categorical, we would not need to do this, because there would already be a version of the variable with one entry per response category per pupil, created by MLwiN when we added the variable to the model, so we could simply use this to plot against. But because our explanatory variable is categorical, longer versions of the dummy variables and not of the variable itself were created by MLwiN to add to the model).

¹not including the reference category

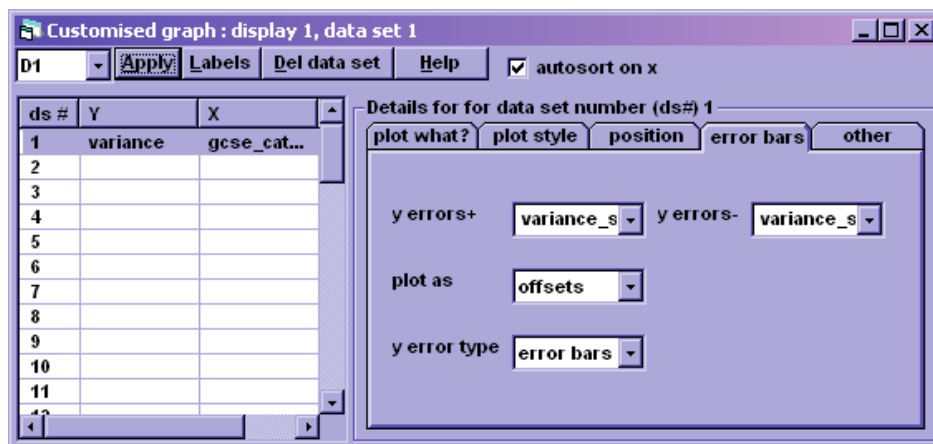
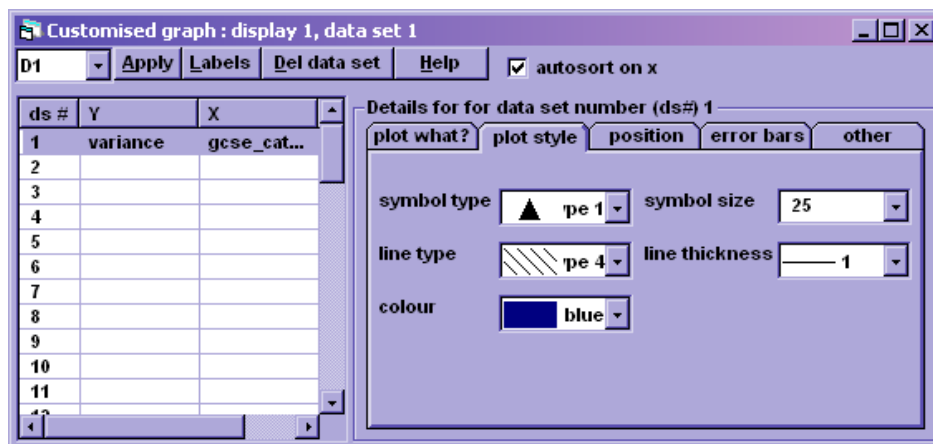
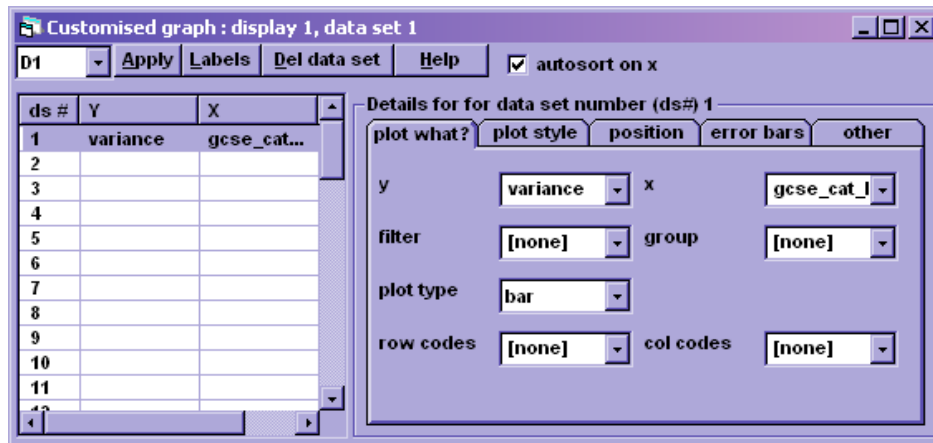
- From the **Data manipulation** menu, select **Merge(replicate)**
- In the **Merge from ID** drop-down box, select **pupil**
- In the **Onto ID** drop-down box, select **pupil_long**
- Under **Input columns**, select **gcse_cat**
- Under **Output columns**, select **c29** (or any free column)
- Click **Add to action list** and **Execute**
- Rename **c29** **gcse_cat_long**

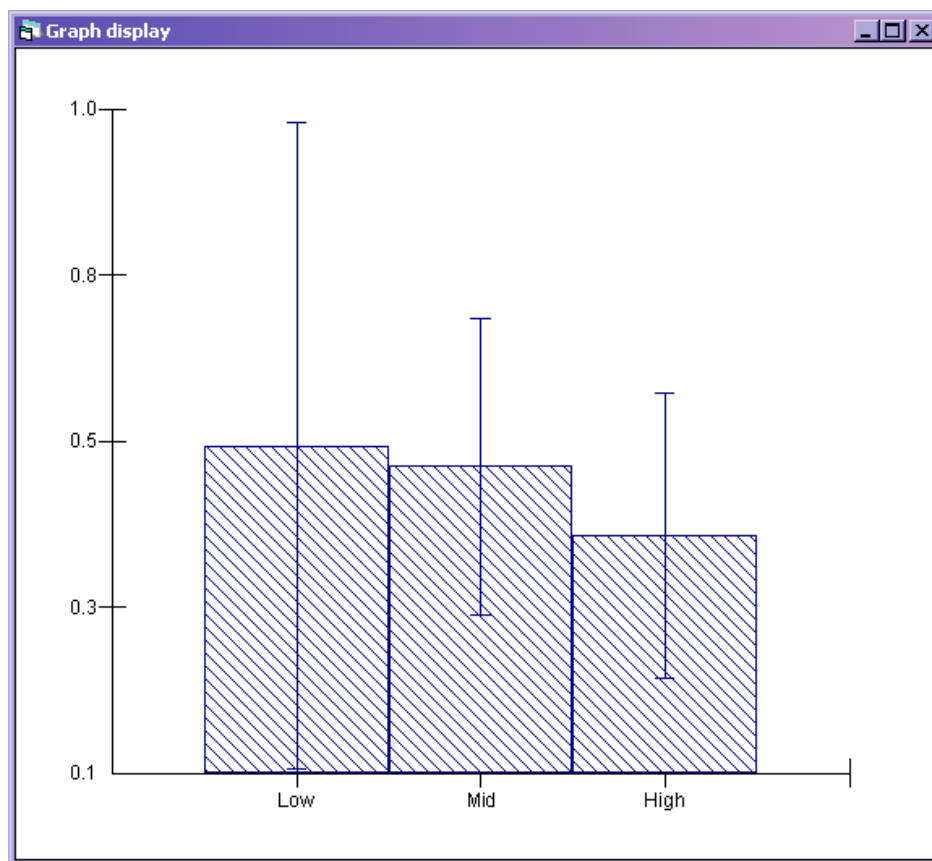
Our original variable **gcse_cat** was categorical, but the newly created **gcse_cat_long** is currently continuous. We can simultaneously declare it as categorical and give it the appropriate categories by pasting the categories from **gcse_cat**:

- In the **Names** window, highlight **gcse_cat**
- Click **Copy** under **Categories**
- Highlight **gcse_cat_long**
- Click **Paste** under **Categories**

We can now plot our graph

- From the **Graphs** menu select **Customised Graph(s)**
- In the **Customised graph** window that appears, from the **y** drop-down box select **variance**
- From the **x** drop-down box select **gcse_cat_long**
- From the **plot type** drop-down box select **bar**
- Click on the **plot style** tab
- From the **colour** drop-down box select **16 rotate**
- Click on the **error bars** tab
- From the **y errors+** drop-down box and from the **y errors-** drop-down box select **variance_se**
- From the **plot as** drop-down box select **offsets**
- From the **y error type** drop-down box select **error bars**
- Click **Apply**





The other facility allows the user to specify combinations of values of the explanatory variables for which the variance is desired. This is done using the grid in the left of the window, under the pane giving the formula. This grid shows a column for each explanatory variable involved in the variance function at the level shown in the **level** drop-down box. In this case, all the explanatory variables included in the model are involved in the variance function². If there were further explanatory variables that were included as fixed effects but did not have establishment level random slopes, they would not appear as columns in the grid.

Suppose that we want to calculate the variance for Low, for Mid, and for High students. We will use 3 rows of the grid to do this. We can type a 1 in each row of the grid in the column **cons.12345**, because **cons** is 1 for all cases:

²That is, if we look at this as the two-level model that it is conceptually, with explanatory variables **cons** and **gcse_cat**. If we consider the model that the software is actually fitting, a three-level model with **resp_indicator** at the lowest level, then instead of explanatory variables **cons** and **gcse_cat** we have explanatory variables **cons.(=<=F)**, **cons.(=<=E)**, ..., **cons.(=<=B)**, **cons.12345**, **Low.12345** and **Mid.12345**, which are all interactions between the explanatory variables **cons** or **gcse_cat** and dummies indicating membership of one or more response categories. In this case, **cons.(=<=F)**, ... **cons.(=<=B)** are included as fixed effects but do not figure in the establishment level variance function.

The screenshot shows a software window titled "Variance function". The main text area contains the following equation:

$$\text{var}(v_{5k} \text{cons.12345} + v_{7k} \text{low.12345}_{jk} + v_{8k} \text{mid.12345}_{jk}) = \sigma_{v5}^2 \text{cons.12345}^2 + \sigma_{v7}^2 \text{Low.12345}_{jk}^2 + \sigma_{v8}^2 \text{Mid.12345}_{jk}^2$$

Below the equation is a table with the following structure:

select	cons.12345	Low.12345	Mid.12345	result
	1.000			
	1.000			
	1.000			

At the bottom, there are controls for "level" (set to "3:estab_long"), "calc", "Name", "Help", "Zoom" (set to "100"), and "Copy". Below these are two dropdown menus: "variance output to:" (set to "c27") and "SE of variance output to:" (set to "c28").

Let's put the Low variance on the first line, the Mid on the second, and the High on the third. For Low students, the **Low.12345** dummy is 1 and the **Mid.12345** dummy is 0. For Mid students, the **Low.12345** dummy is 0 and the **Mid.12345** dummy is 1. For High students, both dummies are 0.

The screenshot shows the same "Variance function" window. The equation remains the same:

$$\text{var}(v_{5k} x_5 + v_{7k} x_{7jk} + v_{8k} x_{8jk}) = \sigma_{v5}^2 x_5^2 + \sigma_{v7}^2 x_{7jk}^2 + \sigma_{v8}^2 x_{8jk}^2$$

The table below now has the following data:

select	cons.12345	Low.12345	Mid.12345	result
	1.000	1.000	0	
	1.000	0	1.000	
	1.000	0	0	

The bottom controls are the same, but the "variance output to:" dropdown is now set to "[none]" and the "SE of variance output to:" dropdown is also set to "[none]".

Finally, click on each row in turn in the **select** column. The variance for that category will appear in the **result** column.

var($v_{5k}^x_{5} + v_{7k}^x_{7jk} + v_{8k}^x_{8jk}$) = $\sigma_v^2_{5x_5} + \sigma_v^2_{7x_{7jk}} + \sigma_v^2_{8x_{8jk}}$

select	cons.12345	Low.12345	Mid.12345	result
	1.000	1.000	0	0.534
	1.000	0	1.000	0.506
X	1.000	0	0	0.415

$\sigma_v^2_{5} = 0.415$

level: 3:estab_long calc Name Help Zoom: 100 Copy

variance output to: [none] 1.0 SE of variance output to: [none]

These results (and the associated values of the explanatory variables) can be copied using the **Copy** button at the bottom of the **Variance function** window, and pasted into Excel for example. They can be copied from Excel and pasted back into MLwiN's **Names** window to become columns in the worksheet. (In the current version of MLwiN, it is not possible to copy from the Variance function window and paste directly into the Names window).

Name	Cn	n	missing	min	max	categorical	description
cons.var	30	3	0	1	1	False	
Low.var	31	3	0	0	1	False	
Mid.var	32	3	0	0	1	False	
result	33	3	0	0.415	0.534	False	
c34	34	0	0	0	0	False	

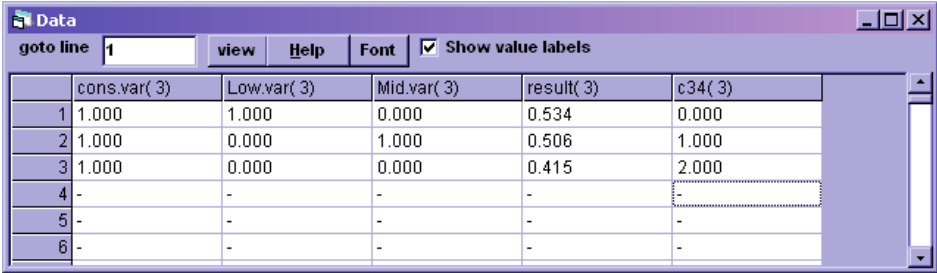
(Here the explanatory variable columns have been renamed, because their original names already belong to other columns in the worksheet).

We can see that these columns have just 3 entries, one per value of the explanatory variable, corresponding to the 3 rows we set up in the Variance function window; they thus store the values of the variance much more efficiently than the 10830 entries we got using the other method.

We might again want to plot a graph using these values. Once again, we have values of the dummy variables here and not of the original variable so we will need to create a version of **gcse_cat** with just 3 rows. (Again, this would not be necessary with a continuous explanatory variable since we would have specified the values of that variable for which we wanted the variance in the Variance function window and these would thus be in one of the columns we have copied and pasted, whereas here we specified the values of the dummy variables associated with our categorical explanatory variable). Probably the easiest way to do this, since there are just 3 rows, is to type in the correct values:

- In the **Names** window, highlight **c30** to **c34**
- Press **View** under **Data**

We will type the values into **c34**. The first row has a 1 under **Low.var**, so this row must have the variance for the **Low** category (and indeed when we were filling in the grid in the Variance function window we decided the first row would be for the **Low** category), so we will type a 0 in the first row of **c34** since that is the number associated with the **Low** category. Similarly, the second row has the variance for the **Mid** category (if we cannot remember that that was what we decided, we can see this because the **Mid.var** variable has a 1 for this row) so we type a 1 under **c34**; and the third row has the variance for the **High** category (by elimination, by memory, or by noting that both the **Low.var** and **Mid.var** dummy variables are 0 for this row so that it must refer to the reference category, **High**) so we type a 2 in **c34**.



	cons.var(3)	Low.var(3)	Mid.var(3)	result(3)	c34(3)
1	1.000	1.000	0.000	0.534	0.000
2	1.000	0.000	1.000	0.506	1.000
3	1.000	0.000	0.000	0.415	2.000
4	-	-	-	-	-
5	-	-	-	-	-
6	-	-	-	-	-

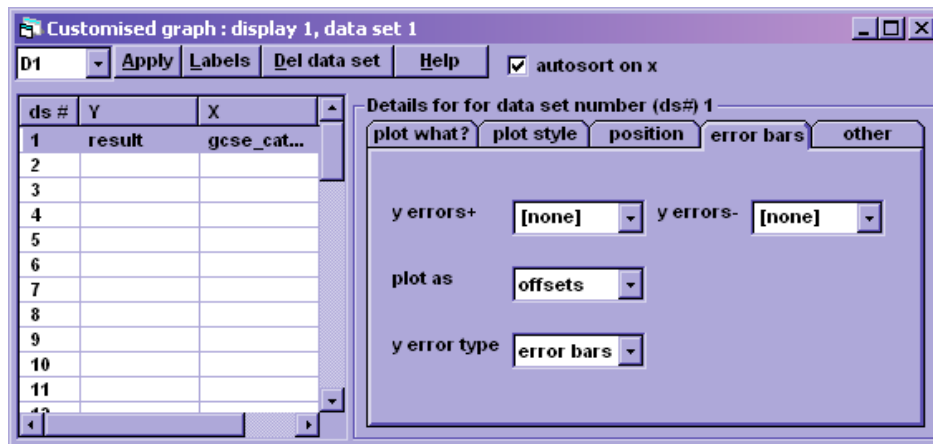
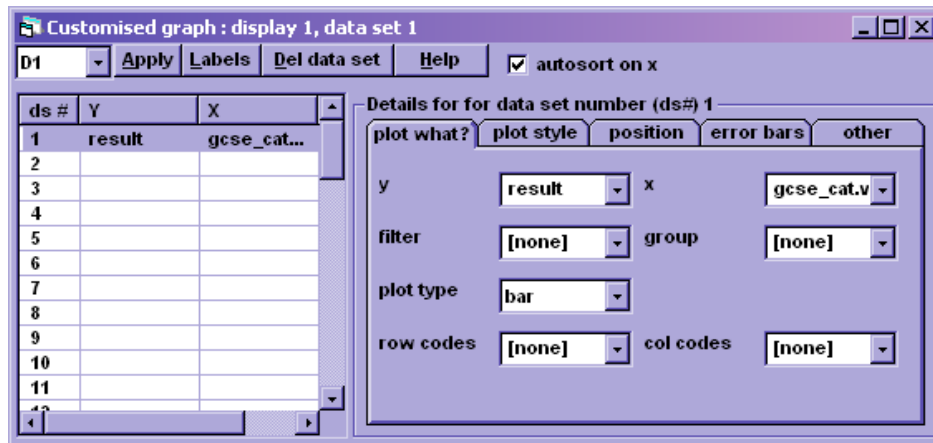
If we had more rows of variances (for example, if there was another variable in the variance function so there were three rows per value of this variable, to give all possible combinations of values of the explanatory variables), then it might be easier to use a command which will put a 0 in c34 wherever the row refers to the **Low** category (**Low.var** = 1 and **Mid.var** = 0), a 1 in c34 wherever the row refers to the **Mid** category (**Low.var** = 0 and **Mid.var** = 1), and a 2 in c34 wherever the row refers to the **High** category (**Low.var** = 0 and **Mid.var** = 0):

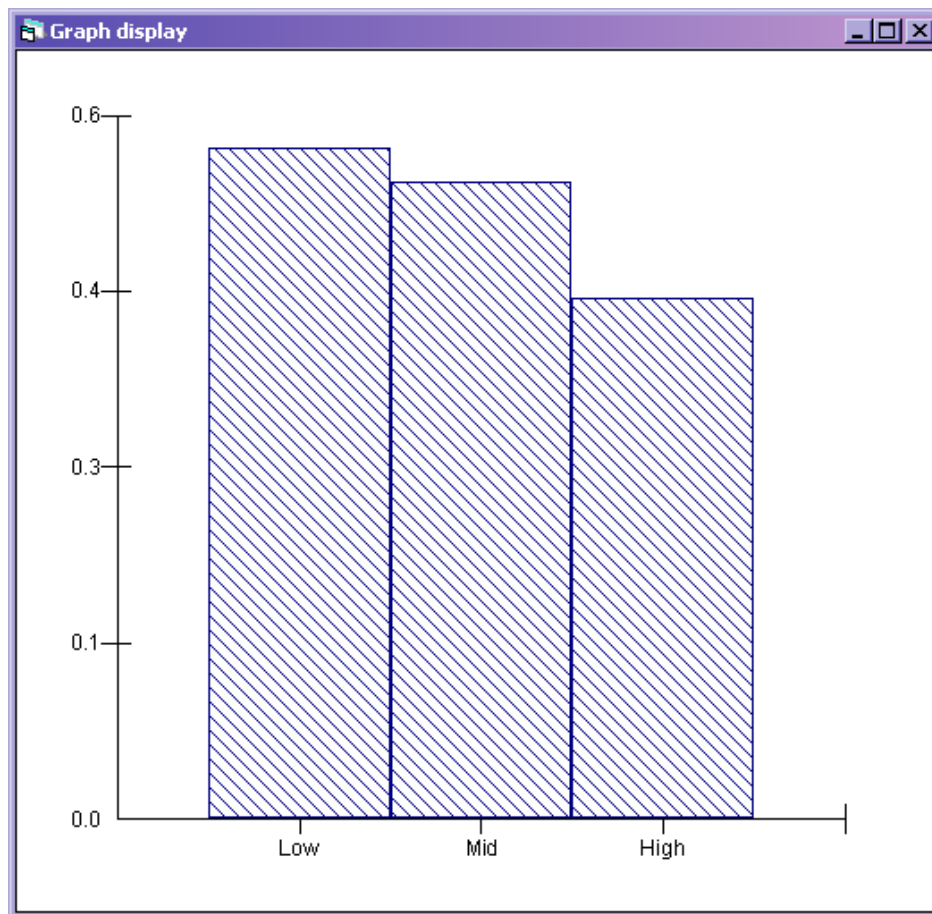
```
► CALC c34 = 2 - 2*'Low.var' - 'Mid.var'
```

We'll rename **c34** **gcse_cat.var**. Once again we'll also need to copy the categories from **gcse_cat** and paste them into this variable.

We can now plot the graph:

- From the **Graphs** menu select **Customised Graph(s)**
- In the **Customised graph** window that appears, from the **y** drop-down box select **result**
- From the **x** drop-down box select **gcse_cat.var**
- From the **plot type** drop-down box select **bar**
- Click on the **error bars** tab
- From the **y errors+** drop-down box and from the **y errors-** drop-down box select **[none]**
- Click **Apply**





A disadvantage of this method compared to the first method is that we do not have standard errors. Apart from this (and the fact that the scale used by MLwiN is different due to not having to fit in standard errors) the graph is the same as the one we plotted earlier.

2 Unordered multinomial models

As mentioned above, using the Variance function window is a bit more complicated for unordered multinomial models than for ordered multinomial models, since (usually) there will be a different variance function for each response category.

We will work with a specially simulated dataset (since in the example dataset for unordered multinomial models supplied with MLwiN, **bang.ws**, the sample size is too small to identify the large number of parameters that results from fitting a model with four response categories each having a random coefficient on several dummy variables). This dataset is called **MCV.wsz**.

Name	Cn	n	missing	min	max	categorical	description
L1ID	1	100000	0	1	100000	False	Level 1 identifier
L2ID	2	100000	0	1	1000	False	Level 2 identifier
cons	3	100000	0	1	1	False	Constant
expvar	4	100000	0	1	3	True	Categorical explanatory variable
response	5	100000	0	1	3	True	Categorical response variable
c6	6	0	0	0	0	False	
c7	7	0	0	0	0	False	

The dataset includes a categorical response and one, categorical, explanatory variable. Again, this categorical explanatory variable is ordinal, with categories **Low**, **Mid**, and **High**.

We start by setting up and running this model:

```

respijk ~ Multinomial(consjk, πijk)
log(π1jk / π3jk) = β0kcons.response_1ijk + β2kMid.response_1ijk + β3kHigh.response_1ijk
β0k = 0.057(0.026) + v0k
β2k = 0.132(0.034) + v2k
β3k = 0.252(0.040) + v3k
log(π2jk / π3jk) = β1kcons.response_2ijk + β4kMid.response_2ijk + β5kHigh.response_2ijk
β1k = 0.070(0.026) + v1k
β4k = -0.081(0.028) + v4k
β5k = -0.161(0.034) + v5k

[ v0k ]
[ v1k ]
[ v2k ]
[ v3k ]
[ v4k ]
[ v5k ]
~ N(0, Ωv) : Ωv = [ 0.585(0.029)
                    -0.300(0.023) 0.580(0.029)
                    0.260(0.028) -0.142(0.028) 0.797(0.051)
                    0.251(0.033) -0.115(0.033) -0.041(0.043) 1.012(0.070)
                    -0.254(0.023) 0.261(0.023) -0.305(0.033) 0.109(0.035) 0.404(0.033)
                    -0.287(0.029) 0.241(0.028) 0.126(0.037) -0.405(0.047) -0.046(0.030) 0.531(0.049) ]

cov(yijk, yijk) = - πijkπijk/consjk : s ≠ r; πijk(1 - πijk)/consjk : s = r;

```

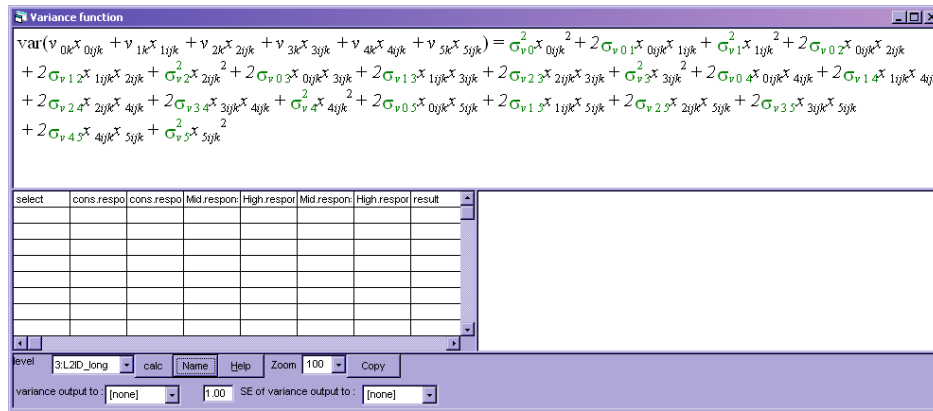
(The reference category for the response is **response_3**; the reference category for the explanatory variable is **Low**; there is a random effect at the highest level on **cons** and both dummy variables, for both response categories).

Although both response categories have **cons** and the two dummy variables in the equation for the variance, their equations for the variance are not the same. This is because the random effects associated with **cons** and with the dummy variables are different for each response category, and accordingly different variance and covariance estimates are involved in the variance function for each response category. For example, the (estimated) variance of the random effects associated with the **Mid** category is 0.797 for response category 1, and 0.404 for response category 2.

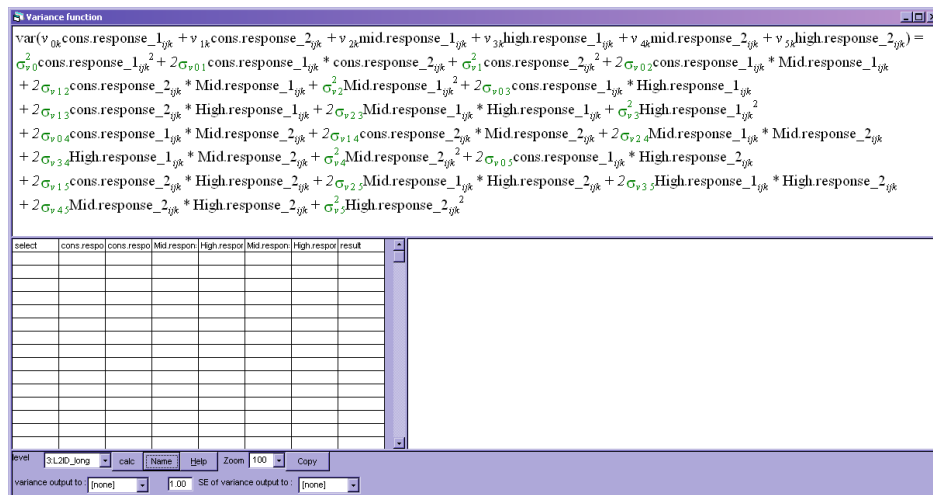
Let us now calculate these variance functions. Once again we have two options: to calculate the variance for every case, or for specified values of the explanatory variables. We will start by calculating the variance for every case.

- From the **Model** menu, select **Variance function**

- From the level drop-down box select **3:L2ID_long**

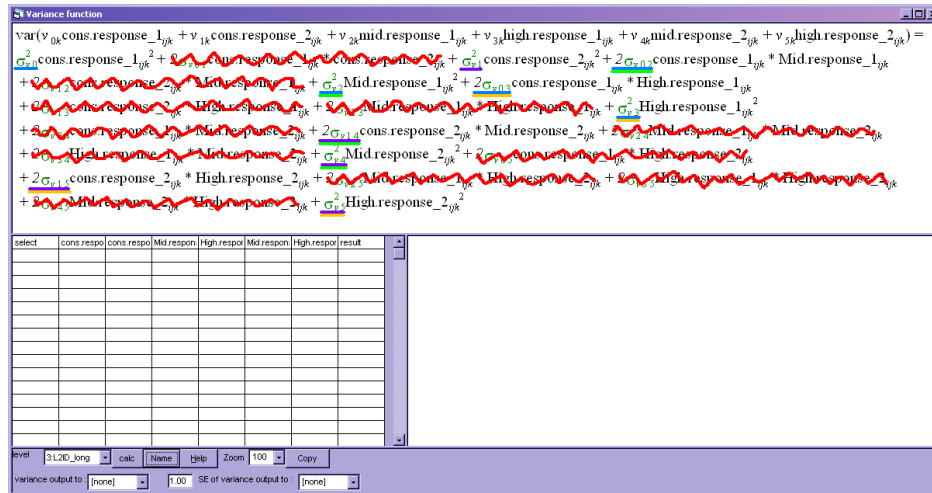


What will be calculated for each case is the horrific formula shown in the top pane. We can make it slightly more intelligible by using variable names instead of *x*s, by pressing the the **Name** button.



Remember that the Variance function window calculates the variance for each lowest level unit of the model fitted in the Equations window, and that in this case the lowest level is **resp_indicator**. Each row refers either to response category 1 or response category 2. For rows referring to response category 1, all the terms ending in **.response_2** are 0 (and vice-versa), because **response_2** is a dummy variable for whether each row refers to response category 2. Some terms will always be 0, for example the second term, $2\sigma_{v01} \text{cons.response}_{1_ijk} \times \text{cons.response}_{2_ijk}$, because either **response_1** = 0 or **response_2** = 0. In addition, at least one out of **High** and **Mid** will be 0 for every case, which means a few more terms in the function will be 0. Finally, remember that all the variables in this equation are either dummy variables or always 1, so the coefficients in the variance function are multiplying either 0 or 1. The coefficients are either the variances estimated in the Equations window, or the covariances multiplied by 2. Thus although it looks frightful, for each case the function simplifies to a sum of at most 3 of the parameters estimated in the Equations window (with some being multiplied by 2). This is shown below, where terms that are always 0 (because they involve both **response_1** and **response_2**, or both **Mid** and **High**) are scribbled out in red, terms which are only involved in the function for response category 1 are underlined in blue, terms which are only involved in the function for response category 2 are underlined in purple, terms which are

only involved for **expvar** = **Mid** are underlined in **green**, and terms which are only involved for **expvar** = **High** are underlined in **yellow**:



We will now calculate the variance.

- From the **variance output to:** drop-down box select a free column (say c18)
- From the **SE of variance output to:** drop-down box select another free column (say c19), and change the 1.0 in front of this to 1.96 (again this step is optional; you may not want to calculate standard errors)
- Click **calc**
- Rename c18 to **variance** and c19 to **variance_se**

We can plot the variances as they are; again we will need to create a long version of **expvar**, with one entry per response category per level 1 unit:

- From the **Data manipulation** menu, select **Merge(replicate)**
- In the **Merge from ID** drop-down box, select **L1ID**
- In the **Onto ID** drop-down box, select **L1ID_long**
- Under **Input columns**, select **expvar**
- Under **Output columns**, select **c20** (or any free column)
- Click **Add to action list** and **Execute**
- Rename **c20** **expvar_long**

Once again we need to paste categorical information from **expvar**:

- In the **Names** window, highlight **expvar**
- Click **Copy** under **Categories**
- Highlight **expvar_long**

- Click **Paste** under **Categories**

We can now plot the graph.

- From the **Graphs** menu select **Customised Graph(s)**
- In the **Customised graph** window that appears, from the **y** drop-down box select **variance**
- From the **x** drop-down box select **expvar_long**
- From the **plot type** drop-down box select **bar**
- From the **group** drop-down box select **resp_indicator**
- Click on the **plot style** tab
- From the **colour** drop-down box select **16 rotate**
- From the **line type** drop-down box select **type 4**
- Click on the **error bars** tab
- From the **y errors+** drop-down box and from the **y errors-** drop-down box select **variance_se**
- From the **plot as** drop-down box select **offsets**
- From the **y error type** drop-down box select **error bars**
- Click **Apply**

This is likely to either crash MLwiN, or result in wrongly plotted error bars (which stretch out of the visible portion of the graph). This is because we are plotting so many points: if we look in the **Names** window we can see that our **variance** column has 200,000 entries (one per response category per level 1 unit in our original dataset). MLwiN's graphing functionality does not cope very well with extremely large numbers of points. We need to reduce the number of points we plot. We can do this without losing any information: recall that our one-entry-per-response-category-per-level-1-unit, 200,000 entry column is storing the values of the variance very inefficiently. The variance depends only on the response category and **expvar**, and so there is one value of the variance for each combination of values of these: in other words, 6 values of the variance. We can thus cut our number of points right down by just keeping one row per combination of values of the response category and **expvar**. First we will create a new variable which looks at the values of **resp_indicator** (which says which response category each row refers to) and **expvar_long**, and takes on a different value for each combination of values of these variables. We do this using a command:

```
► COMB 'resp_indicator' 'expvar_long' c21
```

We will rename c21, containing this new variable, **combination**.

We can see how this variable indicates the combination of **resp_indicator** and **expvar_long** if we look at these variables in the **Data** window:

	resp_indicator(200000)	variance(200000)	variance_se(200000)	expvar_long(200000)	combination(200000)
1	response_1	0.585	0.058	Low	response_1Low
2	response_1	0.580	0.057	Low	response_1Low
3	response_1	1.901	0.190	Mid	response_1Mid
4	response_1	1.505	0.155	Mid	response_1Mid
5	response_1	2.099	0.227	High	response_1High
6	response_1	1.594	0.185	High	response_1High
7	response_1	2.099	0.227	High	response_1High
8	response_1	1.594	0.185	High	response_1High
9	response_1	1.901	0.190	Mid	response_1Mid
10	response_1	1.505	0.155	Mid	response_1Mid
11	response_1	0.585	0.058	Low	response_1Low
12	response_1	0.580	0.057	Low	response_1Low
13	response_1	1.901	0.190	Mid	response_1Mid
14	response_1	1.505	0.155	Mid	response_1Mid
15	response_1	0.585	0.058	Low	response_1Low

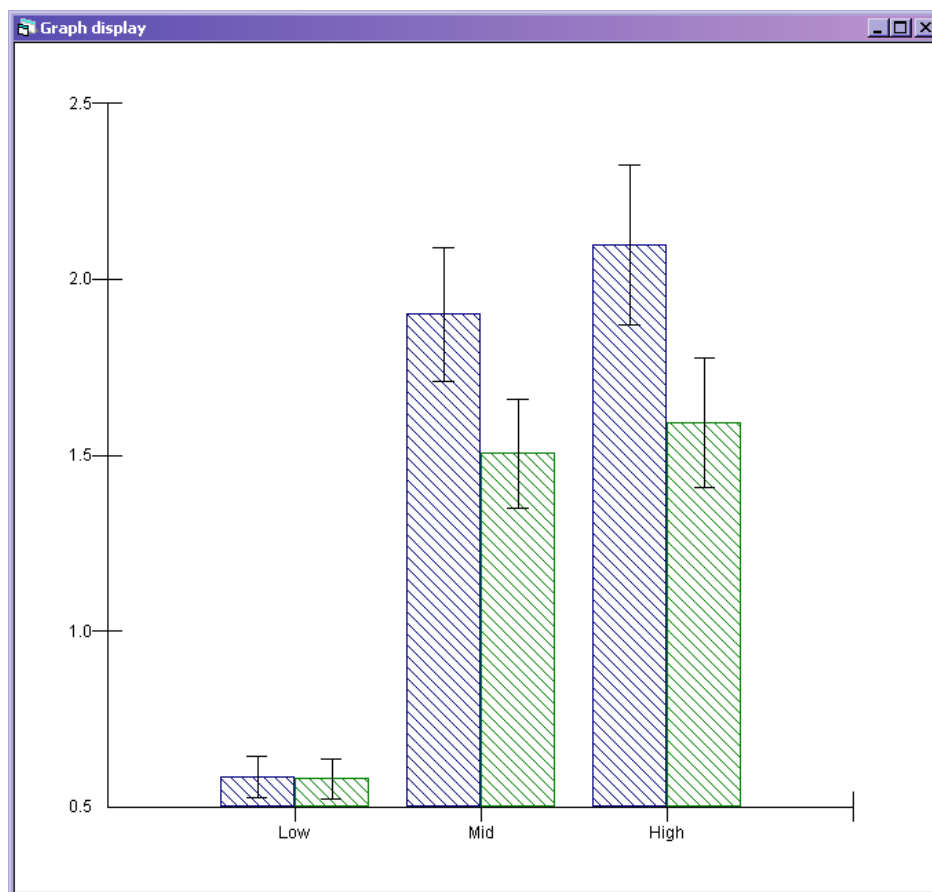
Next, we will sort these variables by our new variable **combinations** so that all rows with the same values of **expvar_long** and **resp_indicator** are adjacent, and then take just the first row from each of these groups of rows with the same combination. (We don't need to keep these variables in their current order, which is sorted by level 1 unit within level 2 unit, since the variance does not depend on which level 1 unit or level 2 unit a row refers to: the variance is the same for all level 1 units and for all level 2 units).

- From the **Data manipulation** menu select **Sort**
- From the drop-down box under **Key code columns** select **combination**
- Under **Input columns**, select **resp_indicator**, **variance**, **variance_se**, **expvar_long** and **combination**
- Under **Output columns**, click **Same as input**
- Click **Add to Action list** and **Execute**
- From the **Data manipulation** menu select **unreplicate**
- In the **Take data** window which appears, from the **Take first entry in blocks** defined by drop-down box select **combination**
- Under **Input columns** select **resp_indicator**, **variance**, **variance_se**, **expvar_long** and **combination**
- Under **Output columns**, select 5 free columns (say c22-26)
- Click **Add to action list** and **Execute**

We will rename the new columns **resp_indicator_short**, **variance_short**, **variance_se_short**, **expvar_short**, and **combination_short**. We'll also need to paste category information into **expvar_short** (and optionally **resp_indicator_short**). Looking in the **Names** window, we can see that each has only 6 entries, as we wanted. We can now have a second go at plotting the graph.

- From the **Graphs** menu select **Customised Graph(s)**
- In the **Customised graph** window that appears, from the **y** drop-down box select **variance_short**
- From the **x** drop-down box select **expvar_short**
- From the **plot type** drop-down box select **bar**

- From the **group** drop-down box select **resp_indicator_short**
- Click on the **plot style** tab
- From the **colour** drop-down box select **16 rotate**
- From the **line type** drop-down box select **type 4**
- Click on the **error bars** tab
- From the **y errors+** drop-down box and from the **y errors-** drop-down box select **variance_se_short**
- From the **plot as** drop-down box select **offsets**
- From the **y error type** drop-down box select **error bars**
- Click **Apply**



We could alternatively use the other functionality of the Variance function window, and directly calculate the variance for the six possible combinations (although we will not then get the standard errors).

If we return to the **Variance function** window, and examine the grid under the pane with the formula, to the left, we can see that there are six columns for us to fill in. However, it is a little difficult to tell what each column refers to, because the ends of the column names are cut off, and the columns cannot be made wider. You can work it out by looking at the order in which the random effects appear in the variance-covariance

The screenshot shows a software window titled "Variance function" with a complex mathematical formula for variance. Below the formula is a table with columns: "select", "cons.respo", "cons.respo", "Mid.respon", "High.respor", "Mid.respon", "High.respor", and "result". The table contains several rows of data, with the first three rows having values in the "select" column and the last three rows having values in the "result" column.

For the remaining spaces, the response category dummy is 1 so what should go in here is just the value of the dummy for **expvar**. The upper left spaces are in columns both referring to response category 1; the first row is for the **Low** category of **expvar** so we put 0 for both columns, the second is for **Mid** so we put a 1 in the first column and a 0 in the second, and the third is for **High** so we put a 0 in the first column and a 1 in the second. Similarly, the lower right spaces are in columns both referring to response category 2; we fill the rows in in the same way.

This screenshot shows the same "Variance function" window, but now the "result" column in the table contains numerical values. The first three rows have values 0.58, 1.90, and 2.09. The last three rows have values 0.000, 1.000, and 1.000.

We can now calculate the variances by clicking in the **select** column of each row

The screenshot shows the 'Variance function' window in MLwiN. The main area contains a large mathematical expression representing the variance-covariance matrix. Below this, a table with columns 'select', 'cons respon', 'Mid respon', 'High respon', and 'result' is visible. The 'result' column contains the value 1.594. The window also shows a 'level' dropdown set to '3L2D_Jong', a 'calc' button, and a 'variance output to' field set to 'c18'.

and copy the grid into Excel (or wherever), then back to MLwiN. The names of the columns will not carry across (apart from **result**) since there are already columns with those names; we'll call them respectively **response_1**, **response_2**, **Mid_r1**, **High_r1**, **Mid_r2**, and **High_r2**. Now we will add two columns, one giving the response category and one giving the value of **expvar**, each in a single variable instead of as dummies. We can do this just by typing in the **Data** window:

The screenshot shows the 'Data' window in MLwiN. The window displays a grid of data with columns labeled 'response_1(6)', 'response_2(6)', 'Mid_r1(6)', 'High_r1(6)', 'Mid_r2(6)', 'High_r2(6)', 'result(6)', 'c34(6)', and 'c35(6)'. The 'result' column contains values 0.585, 1.901, 2.099, 0.580, 1.505, and 1.594. The 'c34' and 'c35' columns contain values 1.000, 1.000, 1.000, 2.000, 2.000, and 3.000 respectively. The window also shows a 'goto line' dropdown set to '1', a 'view' button, and a 'Show value labels' checkbox checked.

Alternatively, we can use a command (which might be simpler if we had many more rows, perhaps because we had other variables also involved in the variance function):

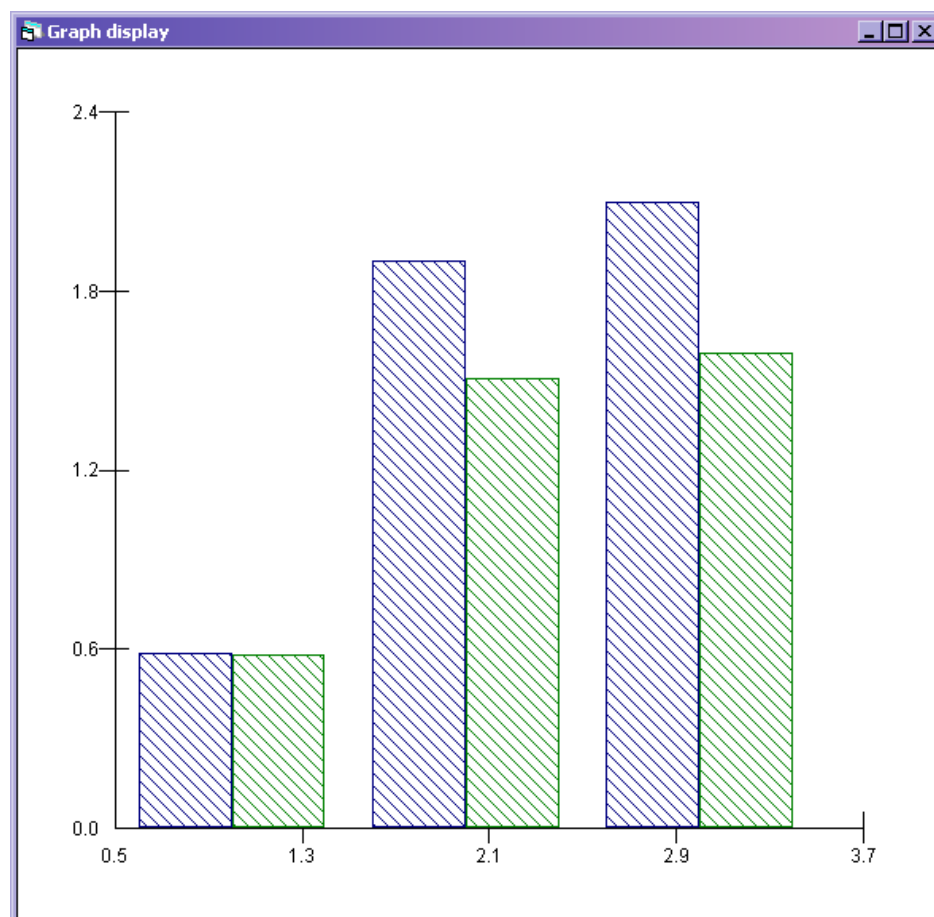
- ▶ CALC c34 = 'response_1' + 2*'response_2'
- ▶ CALC c35 = '1 + 'Mid_r1' + 2*'High_r1' + 'Mid_r2' + 2*'High_r2'

We can name these new variables **resp_indicator_short2** and **expvar_short2**

Now we can plot our graph

- From the **Graphs** menu select **Customised Graph(s)**
- In the **Customised graph** window that appears, from the **y** drop-down box select **result**
- From the **x** drop-down box select **expvar_short2**
- From the **plot type** drop-down box select **bar**

- From the **group** drop-down box select **resp_indicator_short**
- Click on the **plot style** tab
- From the **colour** drop-down box select **16 rotate**
- From the **line type** drop-down box select **type 4**
- Click on the **error bars** tab
- From the **y errors+** drop-down box and from the **y errors-** drop-down box select **[none]**
- Click **Apply**



Again, we have the same graph that we produced using the other option, but without the standard errors.