# MicLog

## 1.1 PREAMBLE

MicLog is a simple to use shareware program for loglinear modelling of multiway tables, whose name, besides betraying its celtic ancestry, is intended to indicate that it was devised for life in a microcomputer. There exist, for such analyses, grander PC programs (such as GLIM) with a more extended repertoire. But within its claimed competences MicLog is: markedly **faster**; possibly **easier** to use (more transparent command sequences - with **menus** - and clearer output); better at handling 'topological' **constructed variables**; and is available **free** of charge as an academic shareware program.

The multiway tables to be analysed can be read in at the keyboard, or written to and read from disk in a variety of formats. A central feature is read/write compatibility with the datasets assembled for James A Davis's program **CHIP**, but MicLog will also accept files in **GLIM** format. Data files can be combined to form new variables, or variables can be deleted.

Models, as is customary, are specified by indicating the sets of 'marginals' to be exactly replicated by the fitted values. There are comprehensive commands for sequentially modifying specified models; user specified ('exogenous') terms can be retained automatically in all models. The fit of models is reported, along with the results of pertinent significance tests; the program can also be set to analyse the contribution of terms within a model. Fitted values are available for inspection. Various auxiliary statistics (such as Hoelter's 'n for rejection') are available as options.

One novel modelling feature is that '**Topological'** constructed variables can readily be defined, assessed, displayed, and amended; they can also be saved on disk for later use. The topologies are defined as concatenations of categories of existing variables. A 'hill-climbing' algorithm is provided for (mindlessly?) maximising the fit of such topologies.

MicLog provides flexible facilities for examining data and residuals - either cell by cell across the full tabulation, or aggregated across specified subtables. Odds-ratios can be displayed, on observed and modelled data, for any specified subtable (and significance tests or ranges are reported). The program also enables recoding of the data (covering also deletion of categories or variables), and modifications to category size whilst retaining the 'structure' (as assessed by odds-ratios) of the data.

Output of results can be copied to files, for direct incorporation into word-processor documents, and there is a 'review' facility, to scan back through the results of an analysis run, to enable informed analyses. Colour screens are supported (though not required).

The one facility omitted you might expect included is report of category-specific 'effect coefficients'. Mainly because, with Holt (1979)[1], we hold that the effect coefficients in interaction models, deriving from uninterpreted constraints, are strictly uninterpretable. Partly because estimation runs faster without them. And we do relax our censure and provide them for 'topological' variables.

The program is **menu driven** – though not yet a fully integrated 'windows' program, and much incidental information is provided on screen. These menus should allow the user to run MicLog without frequent recourse to this manual.

---

[1] 'Loglinear models for contingency table analysis: on the interpretation of parameters' *Sociological Methods and Research* **7**:330-336. But see also J. S. Long (1984) 'Estimable functions in log-linear models' *Sociological Methods and Research* ***12***:399-432.

## 1.2 Size restrictions.

The standard size MicLog accepts up to **7** observed **variables**[2], allows a further **7** topological variables in use at any one time, and handles integer or real data with up to **5,000 cells** in a table.

Larger versions of MicLog are available relaxing these constraints. Various combinations of these maxima (variables, cells, topological variables) are possible, so if standard MicLog is too restricted for your purposes, let me know.

## 1.3 Menus

The program is menu driven, and much incidental information is provided on screen. Menus allow - should allow - the user to run MicLog without frequent recourse to this manual. Notice though that menus are context sensitive - for example, when you first enter the *Data* menu it invites you only to *read* data, but after you have read data it offers options to inspect and amend the data. The task of this manual is to give some sense of the possibilities of the program, and to give detailed guidance on model specification. Detailed input instructions are given in this manual only when judged necessary; otherwise you just invoke the appropriate menu in the program and it should adequately guide you. You can always escape from a menu without taking action (by pressing plain [Enter]), so do explore.

> *As a typographical convention in this manual* we deploy the convention of here writing the menu calls separated by slashes to indicate sequential menu invocation, starting from the base menu. Thus, if you are referred to the *Data**Define* menu this asks you to select *Data* from the *Main* menu, and *Define* from the menu which ensues.

## 1.4 Experienced users avoid chat

*[Read this later.]*

Chatty programs can be a pain; MicLog provides two attenuations of its loquacity for the experienced; they can be combined.

**a**. The explanatory chat (that tells you '*what a topological variable is*', and such) can be silenced, *whilst retaining the menus*, by invoking the *Switches**Chat* option. In silenced mode, if starkly asked for a menu choice, or a filename, or a number, or whatever, you can reply:

```
?
```

and MicLog will produce all the text that would otherwise have preceded the request in question.

b. Menus (and any presenting chat) can be avoided altogether by typing ahead from the base prompt (the symbol '»'). Type the pertinent key-letters, and any parameters, separated by space(s). So, on entry, when the *Main* menu demands *Choose*, you might specify:

```
d g  mydata
```

to get, with no admonitions, the data in file *mydata.doc*. Once models are input, MicLog refrains (unless requested) from repeating the main menu, but all its commands remain available. Thus the experienced user need never leave model input mode; in that mode she might type:

---

[2] Each variable is allowed up to 35 categories, but only up to 15 categories are allowed to be active in any one analysis.

```
*2
r f acd
```

to fit the model with all-two-way terms, and look at the full residual report for table ACD, all with no MicLog exhortation. If you do not complete a command-sequence then MicLog will resume its prompting; for example, type

```
r f
```

and MicLog will tell you about 'full residual' layout, remind you of the 'table specifying' conventions, and request you to *Specify the table*. If 'chat' has been suppressed (see above) you will only see the demand to *Specify the table* (but typing **'?'** in reply would elicit the full accompanying chat for this item).

## 1.6 Direct Output

*Screen control*: MicLog tries to split the screen display into sensible 'pages'; you can modify the number of 'lines' shown by the *Switches**Screen* menu.

*Printing:* For serious transcription - incorporating results in an article or essay, and for monitoring the progress of a complex analysis - see *Section 1.7* on sending results to a file.

## 1.7 Output to a word-processing file

Miclog can be set to echo its results to a file; you can then incorporate these results directly, using your word-processor, into whatever masterpiece of prose you are composing. Direct incorporation saves time and errors. The output file can also be interrogated from within MicLog, giving a **review** facility (see next section).

The *Word* menu controls the output file; you can either open a *new* file or *append* the current results at the end of a pre-existing file. The menu will prompt you for the name of a file; by default[3] the file will be in your current directory (but you can, if you wish, precede the filename by a full DOS directory address); by default the file will be given the extension DOC (but you can explicitly specify an extension). At present MicLog understands only short (eight character) file and directory names.

In *Word**New* if the file you specify exists (and so would be overwritten), MicLog checks with you before overwriting (unless you use the filename TEMP, which MicLog regards as indicating a disposable file).

In *Word**Append* (which appends the output to the end of a pre-existing file) MicLog first checks that the file was a MicLog output file (if not, it asks for confirmation before proceeding).

In either case the output is a plain ('unformatted') text file, written in 80 character lines; when printing with your word-processor it is best to arrange for the incorporated tables to appear in some *non*-proportional font (to preserve table layout).

When output is echoed to a file, only substantive results are echoed (the menus and such[4] do not appear); but any command line which begins with a full-stop[5] is read as a **comment** and copied, unprocessed, to the output file, to allow notes to your future self.

---

3 The default location for text files can be reset from the *Switches**X**W* menu.

4 The \Word menu provides an option to echo the *command* input lines - useful for teachers making demonstration notes?

5 My mnemonic to self was '*point* to note'

## 1.8 The view/review facility

When output is sent to a file, MicLog allows you to read and page through that file without leaving the program. This can be very useful in keeping track, without the need to have recourse to pencil and paper, of a complex analysis. So, even if you do not intend to incorporate the results, there may be merit in routinely[6] opening a *Word* processing file (the special treatment of the name TEMP - see above - was intended to enable the command sequence **w n temp** to always open such a scratch file with minimum hassle)

The *View* menu, when invoked, by default displays the last screen in the file. It then provides commands to skip to the *top* or the *end* of the file (positive page numbers on the display count from the top, negative from the end, of the file), to page through the file in either direction (by *previous* and *next*), or to *jump* to a specified screen page (using the positive or negative[7] numbering convention). These options can, as usual, be specified when the command is invoked; thus:

```
» v j 5
```

would jump again to the interesting results you had noted on page 5, to compare to your current model.

*Hint:* If today's results are being sent to a file named TODAY and I suddenly feel the need to examine my *previous* analyses in the file YESTERDY, then the sequence:

```
w a yesterdy
```

will open that file (closing TODAY), then *View* will allow me to page through YESTERDY, and when curiosity is satisfied, the sequence:

```
w a today
```

will reset output to continue today's results from where they were suspended. And I can obviously inspect *them* as required.

## 2.0 MODELLING

To specify a model, you (obviously) first input data (see *Section 5*) and then (in response to the main '»' prompt symbol) simply type a model (see next section).

After estimating a model, MicLog will then prompt for the next instruction (again with the '»' symbol) but without repeating the *Main* menu. The options of the *Main* menu remain directly available, and you can reinvoke its listing by typing simply:

```
M
```

To obtain guidance on the syntax of model specification, type:

```
?
```

When it first suppresses the *Main* menu listing, MicLog reminds you of the availability of these 'M' and '?' commands, and a null response (the plain **[Enter]**) to the '»' prompt will also elicit that reminder, so the 'invisible' *Main* menu should create no problems. Experienced users will probably drive the program throughout without invoking the *Main* menu (see *Section 1.4*).

---

6 Fortran read/write routines are deathly slow, so perhaps this routine makes sense as *routine* only on hard-disk installations.

7 If an output file is *active*, only *positive* page numbers, which count from the *top* of the file, will be stable across calls to *View*.

## 2.1 Model specification

Loglinear modelling on crosstabulation can be regarded, as its simplest, as asking *which sets of observed marginals do we need to know in order to replicate the full tabulation*. For the estimated values, the program starts with exactly-equal cells, and iteratively constrains these to replicate the selected marginals of the model; the estimated values have no structure other than that arising from the imposition of these marginals. These estimated values are then compared to the full observed data.

Variables are identified by single letters[1]; thus, if the full tabulation is of variables ABCD, and the model examined is:

```
[AB][AC][D]
```

this would be to ask, *can we adequately reproduce the ABCD data just knowing the [AB] table, the [AC] table and the [D] table* (these tables are marginals of the [ABCD] table).

The usual pattern of analysis is that the user will specify some base model, examine various modifications to the base model, choose one of these as the fresh base. Possibly iterate. The modifications examine the information that is provided by knowing, or lost by removing, certain marginals when certain other marginals are known. The information provided, or discarded, with a term is always model-specific knowledge.

To ease this task, you can designate a set of interactions as *exogenous* to your current explanatory puzzles, and MicLog will then always include these in any model without further intervention from you (see *Section 2.5.1*).

## 2.2 Base models

Base models can be specified in standard notion, using round or square brackets, and no embedded spaces[2], thus:

```
[ AB] [ C]
```

The $G^2$, with the associated 'degrees of freedom', will be calculated, its 'significance' reported (and optional PRE and 'n for rejection' figures calculated). The fitted values are calculated, and available for further analysis (see *Sections 3.2, 3.3*)

That, standard, model syntax is cumbersome (for the square brackets are serving merely as separators, not brackets). so MicLog lets you use *comma* as a singleton separator to obtain the shorter[3] representation:

```
AB, C
```

If your model specification contains 'redundant' marginals, MicLog will prune the model appropriately.

Thus: `ABC,BC,D` is seen as: `ABC,D`

With several variables, typing, say, all the three-way terms can be tedious (and for me error prone). So to aid specification of the typical *all n-way terms* base model we have the shorthand command:

```
*n
```

---

1 As explained below, the user can allocate letters with some mnemonic value to the variables (the defaults are A,B,C, *etc*).

2 Should a model be about to overflow a line, type the character '&', and continue, with no embedded spaces, model input at the start of the following line; thus:

```
ABC,D&
C,BD
```

is read by MicLog as 'ABC,DC,BD'

3 Improbable single-term models involving less than two variables may, if written in compact syntax, be read as other MicLog commands; for them use the full splendour of brackets.

where n is some integer.

Thus, for 4-way data [ABCD], the command *2 is interpreted as
AB,AC,AD,BC,BD,CD

Note that with the *Exogenous* option in force (see *Section 2.5.1)* any 'exogenous' terms defined will be automatically included in any base.

Thus, for 4-way data [ABCD], with [ABC] prespecified as an 'exogenous' term, the command *2 is interpreted as requesting the model ABC,AD,BD,CD

A variant of the 'all n-way terms' specification, of particular use when substantively the analysis concern is with the interactions surrounding one specific variable, is:

*n~y

where **y** is some specific variable. This generates a model which includes **y** itself, and includes all *n-way* interactions *other* than those involving **y**.

Thus, for 4-way data [ABCD], where D is 'depression', and ABC are three possible 'causes' of depression, the command *2~D gives: AB,AC,BC,D as the base against which you can explore which two-way interactions with 'depression' matter

### 2.2.1 PRE measures

Comparison of models in terms of $G^2$ may not always be transparent. Proportional Reduction in Error (PRE) measures are common in data analysis ($R^2$ is the most familiar example), and in fitting marginals some find that a PRE figure helps comparisons across models and data. If we have some reference point 'original' error *OE* from some minimal starting model, and also a 'remaining' error *RE* figure from $G^2$ for the model under consideration, then:

PRE = (*OE* - *RE*) / *OE*

You have to choose an apt model for *OE*; one school of thought says '*all simple marginals*'; another prefers '*marginals PLUS exogenous interactions*'. Either way, all you get is a simple linear transformation of $G^2$, and moreover one which you could readily calculate by hand (*i.e.* calculator). Still, MicLog stands willing to do the work for you - invoke the *Switches**E* menu. It then also calculates a PRE measure based on the $G^2/df$ ratios - an oddball index purporting to report proportional reduction in 'noise per degree of freedom remaining' (*i.e.* a measure that takes $OE=G^2/df$, and similarly for *RE*; so this coefficient may be negative whilst unmodified $G^2$ decreases). PRE's are not reported when negative.

This is more prose than these measures are worth. The indices are simply there as possible helps in deciding when/where to be excited. Perhaps use in conjunction with the *Exogenous* option - see *Section 2.5.1*

### 2.2.2 Hoelter's CN statistic

To aid interpretation of the import of particular $G^2$ values, J. A. Davis suggests using Hoelter's **CN** statistic (*Sociological Methodology & Research* 1983), a '*N for rejection'* figure, which transforms $G^2$ onto a "sample size" scale.

For any given $G^2$("*x*"), if the p=.05 criterion value is "*y*" for the given "df", and sample is "N", then:

**CN**, for p=.05, = $y \times N / x$

Crudely, this gives the sample size at which, "all other things being[4] equal" we would have rejected the model (at p=.05). MicLog will, if requested by the *Switches**N* menu, report **CN**, and express it as a proportion of N.

---

4 It is of course understood that this counterfactual does not hold.

CN can be compared both to the sample N and to “standard” (SN) sample sizes (Davis notes around “375” for SN in current leading journals, but subfields vary). Some ranges are more readily interpretable than others. If CN<N<SN then we are probably justified in claiming that model rejection is not a function of “unusually large” sample. But in interpreting CN>N remember that the counterfactual is unlikely to be met: real extra cases need not match the distribution of present cases. Still, used delicately, CN can, as Davis demonstrates, provide extra leverage on problems posed by sample-size.

## 2.3 Adding or deleting terms

We can proceed to modify a given base model by issuing a specification preceded by a plus or minus sign ( **+** - ); this requests terms to be added to or deleted from the base model. One modification may list several terms to be added (or deleted). The resulting modified model is reported (indented, to distinguish it from the base) and the decrease or increase in $G^2$ is reported, its significance calculated; a final figure gives (as a rough measure of the ratio of improvement to information) the $G^2$ shift divided by the shift in degrees of freedom. Fitted values are calculated and are available for analysis.

Subsequent modification commands are *not cumulative* but continue to refer to the current base model (in the next section we see how to redefine the base). The simple command:

**B**

will remind you of the current base model.

As illustration of the non cumulative nature of trial modifications, the sequence:

```
ABC,D
+E
-D
```

would estimate, in sequence:

```
ABC,D
ABC,D,E
ABC
```

There is one 'clever' feature. The deletion command, in an attempt to capture normal intentions, will, when *deleting* an *n*-way interaction, *insert* any of the implied (*n-1*)-way interactions which were not otherwise present in the model. MicLog assumes you wish to remove only the higher order effects.

Thus:

```
AB,C,D
-AB,D
```

would estimate[5]:

```
A,B,C
```

Clever programs can sometimes be a nuisance, so if you intend your deletion request to be interpreted literally use the *\Switches\Algorithm* menu to set (or later unset) the interpretation appropriately.

With *Literal* chosen as the interpretation, the sequence:

```
AB,C,D
-AB,D
```

yields simply [C]

Note that with the *\Exogenous* option in force (see *Section 2.5.1)* no command to delete 'exogenous' terms will be obeyed.

## 2.4 Redefining the base model

A 'slash' issued as a command replaces *the base model* by *the most recently estimated model*. So if we have tried some modification and wish now to take that as our *base* model, we simply type

---

[5] Since deletion proceeds sequentially, the odd looking modifier *-AB,A* could have been used to estimate *B,C*

/

The '/' can also appear at the end of a modification command, with the same effect.

Thus the sequence:

ABC,D
+E/
- D

estimates: *first* ABC,D *then* ABC,D,E *and* (because of the terminating slash) deploys that as the new base, to *then* give:

ABC,E at the final deletion.

## 2.5 Search facilities?

The program does not incorporate any of the (many) competing algorithms suggested for locating 'best' models, since typically the 'optimum' model will be *under*determined by the data; better to explore models connected to particular substantive narratives and queries? But MicLog *does* provide facilities for mechanically maximising the fit of topological models (see *Section 4.7* below) - an unusual feature this.

And MicLog *does* provide some aids for gentle exploration of orthodox models; these are simply commands to 'try *several* modifications', which leave decision-making with the user. The program will, if asked, look at *all possible deletions from* or *all possible additions to* a specified model. And, as a further, low-level, aid, the program can be asked to restrict its attention to models retaining specified 'exogenous' terms.

### 2.5.1 Exogenous terms

The estimation procedure used by MicLog is, of course, neutral between variables - the 'error' minimised is *'error in estimating the cells in the full table'*, **not** *'error in predicting one specific variable'*.

Nevertheless, in selecting terms to examine, you may be guided by some explanatory concern for a variable which substantively you regard as 'dependent'. Suppose we have [ABCD] and are interested in the determinants of variable *D, 'Depression'*. It may be reasonable to assume we know [ABC][D] - the aggregate distribution of 'Depression'; and the full interactions amongst the other variables. We can regard that information as exogenous to our current explanatory concerns. We then ask: '*Which interactions with Depression are additionally required to fit the data?*'

To help keep track of such questions, MicLog lets you specify a set of terms as 'exogenous': these will thereafter *always* be included in any model, and will never removed by any of the 'deletion' commands. With, say, '`ABC,D'` specified as exogenous, the command:

» DB

is read as '» `ABC,DB'` and a subsequent:

» -

would only cut [DB], leaving [ABC][D] intact. The facility (invoked by the *Exogenous* option in the Main menu) adds nothing new; it merely saves computation and output by restricting deletions considered, and input of models, to 'pertinent' cases.

Notice that rather than permitting ourselves knowledge of *all* the interactions amongst the 'explanatory' variables there may be a case (in terms of parsimony and intelligibility) for restricting ourselves to a certain level of interaction. So, continuing our example, it might make sense to specify the 'exogenous' term as '`*2~D'`, giving '`AB,AC,BC,D'` as our base and then exploring which associations with D are, against that background, informative.

*Hint:* It may well make sense, after specifying the 'exogenous' terms through the *Exogenous* command, to take that minimal model as the PRE reference point by next typing:

```
» s e y
```

(see *Section 2.2.1*)

### 2.5.2 Sequential deletion

The subtraction command:

```
-
```

without specific term, will scan through the current base model, deleting each term in turn and comparing each of the truncated models to the base model. (See also *Section 2.6* for an automatic version of this.)

The subtraction command, in an attempt to capture normal intentions, will, when *deleting* an *n*-way interaction, *insert* any of the implied (*n-1*)-way interactions which were not otherwise present in the model. This rule is the same (for the same reasons) as that for the *specific* deletion command (*Section 2.3* above), and is likewise under the control of the *Switches**Algorithm* menu.

Note that with the *Exogenous* option in force (see *Section 2.5.1)* attention is restricted to the subset of models which retain the specified exogenous terms.

### 2.5.3 Sequential addition

The '`+`' equivalent of the '`-`' command would be too large a blunderbuss if it simply requested '*insert each of the omitted terms in turn*', so we specify the level of interaction to be scanned, thus:

```
+*n
```

This command estimates the set of models formed by adding, in turn, each of the omitted *n*-way interactions to the base model. For each constructed model the usual comparisons to the base are calculated.

There is a further 'clever' feature here. MicLog assumes that the current omission of any *(n-1)*-way interaction in the base model is witting and intended, so does not explore any *n*-way interaction which would introduce a previously *absent* lower-level interaction.

For example, suppose the baseline model is:

```
AB,BC,AC,D
```

then MicLog if asked to consider all 3-way interactions, by the command:

```
+*3
```

will consider the addition of [ABC] only. Were it to consider, say [ABD] that would be equivalent to introducing (among others) the interaction [BD]. MicLog assumes you do not wish to include that 2-way term (else you would have explicitly included it amongst your 2-way interactions before moving on to consider 3-way interactions).

This is not a very clever feature. It *may* make sense if you have some strict notion of permitted degrees of complexity in your model - either from some notion of interpretability or parsimony. But, on the other hand, we know that the failure of a 2-way term (such as [BD]) to do useful explanatory work tells us *nothing* about the potential utility of 3-way terms that might subsume it; so we might well wish to introduce such a term?. If you take this latter view, the *Switches**Algorithm* menu can constrain MicLog to a literal interpretation of your commands.

## 2.6 Contribution of terms to the fit of the model

The assessments returned by the sequential deletion command (*Section 2.5.*2) can be seen as equivalent to the assessments provided for individual terms in a regression equation by the individual *t*-tests or[6] *F*-tests. The individual *F* is a linear function of the decrement in the residual sum of squares that would result from adding knowledge of a particular variable to an equation already containing the other variables. The associated probability allows us to assess whether the additional knowledge is significantly different from zero.

Equally we could look at each interaction in our present models and ask whether, *if we knew the other marginals*, that particular marginal made a significant contribution to knowledge. Suppose we were considering a particular three-way term: the question would be about the information contained specifically in that level of interaction, so we would assume knowledge of the implied two-way interactions, as well as knowledge of the other terms in the model.

*Specific analysis*: After any model, the general subtraction '`-`' command (with its deletion algorithm left at the default, 'Intelligent', setting) would yield the required information.

*Automatic analysis:* If you anticipate always requiring such a termwise assessment, MicLog can be set (using the *Switches*\$G^2$ menu) to automatically report the assessment[7] information in tabular form for every model.

*Note* In contrast to a regression procedure, the required statistics are not generated as a by-product of fitting the model; they require separate estimations of separate models. So this option can take time.

## 3.0 EXAMINING THE DATA

You can look either at cell values, or, using the odds-ratio facilities, at the patterns of relations between cells. And the cells can be variously defined. These options (invoked from the *Data* and *Residuals* menus) present data aggregated and arranged as requested; the options will prompt for a particular table, which can be a subset or aggregation of the full data.

Thus, on data [ABCDE]:

```
BCDA
```

specifies a set of BC tables within categories of D within categories of A (and *aggregated* over E).

A table specification can additionally contain reference to particular categories of a variable: you qualify the variable letter by an integer (the category number).

Thus (with *no embedded spaces*):

```
B6CD3A
```

will specify a set of tables showing the distribution of '*category 6 of B' against C,* displayed within '*variable D category 3'*, within the categories of variable A.

The relevant menus will, as usual, provide full guidance on the syntax which, being compressed, may be not easy to *read*, but is straightforward and unambiguous to type.

Note that, using this facility, we can display summary tables which align comfortably

---

6 The square of *t* is *F*

7 Obviously no $G^2$ assessment is undertaken for terms specified as 'exogenous' (see *Section 2.5.1*)

within the confines of a screen display, or present together for comparison particular slices of our data.

## 3.1 Crosstabulation of data

Crosstabulations can be generated from the *Data**Full*, *Data**Numbers*, *Data**Percentages* options, yielding respectively 'data numbers *and* row percentages', 'numbers', 'row percentages'. The point of these last two options is that a wider area of table can be on display, and thus scanned, at one time.

For example:

```
d f BCA
```

would give the table of B by C within categories of A (row percentages are reported within the B by C table, but no other derived statistics are reported).

```
d n B
```

would give the marginal distribution of B.

```
d p B3
```

would give 'variable B category 3', expressed as a percentage of total cases. (You of course need not remember the '`d n'` *etc*. sequences; just invoke the *Data* menu).

## 3.2 Crosstabulations of residuals

The *Residuals* menu (accessible from the *Main* menu once you have estimated a model) allows various inspections of the residuals from the last-fitted model (and it will display the model to remind you). **Read the note, in** *Section 3.2.1***, on precision.** The residuals options will, like the data display options, prompt you for the table within which you wish to see the residuals - and remember the 'category' specification features which make it easy to scan particular residuals.

Thus if you specify the table '`A2B4DC'` the program will show you 'category 2 of A and 4 of B' scanning across D and C. Again, the command is easier to use than it is to read.

*Residuals**Raw* option: this takes the cell-by-cell residuals calculated over the *full* data and reports them (aggregated if need be) within the table you specify.

If *disaggregated* the residuals will be 'signed' (showing positive and negative values);

if *aggregated* their *absolute* values would be taken (so that -1.5 and 0.5 aggregates to 2.0).

MicLog will also report 'the percentage of cases misclassified *by the model*' - this is the percentage of cases taken over the *full* data[1]. This, being an absolute difference measure, not a relative one, is not minimised by the fitting technique (so its rank order across models does not correspond to the rank order of $G^2$ across models), but some find it aids interpretation.

*Residuals**Standardised* option: this again starts from the cell-by-cell residuals across the *full* model, 'standardises' them, where the standardised residual for a cell is deemed to be:

(*Observed - expected*)/sqrt(*expected*)

[1] 'percentage of cases misclassified' is simply the sum of 'the absolute cell-by-cell residuals calculated on the full data' divided by two and expressed as a percentage of the total cases.

and reports them (aggregated if need be) within the table you specify. If disaggregated the standardised residuals will be 'signed'; if *aggregated* their *absolute* values would be taken.

*Residuals**Full* option - a bulkier display than the others. It again starts from the cell-by-cell residuals across the *full* data, but its presentation varies depending upon whether your chosen display table was exactly fitted by the model which generated the residuals:

If the requested table is *not fitted* by the model (*e.g.* you look at `AC` having estimated model `AB,BCD`) then MicLog presents the observed values, the expected values, and (as in *Residuals**Standardised*) the standardised residuals. As in the other options, if the table is an *aggregate* one, these values will be obtained by *summing* absolute cell values from the *full* data (so the 'standardised residual' figure is *not* a measure of the discrepancy between the *aggregate* observed and the *aggregate* expected values).

MicLog will also report 'percentage of cases misclassified *in that table*' which is *not* a measure of the overall percentage of cases misclassified at the level of the *full* data, but a measure of the percentage misclassified[2] *at the level of aggregation* and *over the range* of the *reported* table (so it is a summary of the discrepancy between the *reported* observed and expected values - contrast the *Residuals**Raw* option which always displays 'percentage misclassified' over the *full* data).

If the requested table is *exactly fitted* by the model (*e.g.* you look at `AB` having estimated `AB,BCD`) the *expected* will exactly match *observed* values; so MicLog reports the aggregate observed values, the aggregated contribution to $G^2$ from each aggregate cell, and that contribution as a percentage of the total $G^2$.

If you want to look at the *summed absolute standardised residuals* for such an exactly-fitted table (*summed* across not-exactly-fitted cells, so they will be non-zero), invoke the *Residuals**Standardised* option.

### 3.2.1 Precision of residuals

MicLog (like most 'loglinear' programs) runs an iterative algorithm to estimate the model. It is however more ruthless[3] than some in deciding when to stop; it stops when it reckons further iterations will not change the value of $G^2$ at one decimal point (on the assumption that greater accuracy of $G^2$ will not affect the substance of your argument, but will use scarce processing time).

Whilst this stopping rule is a good one when comparing summary $G^2$ values, it can return slightly imprecise cell-by-cell estimates of the residuals[4]. These are marked by a '$\approx_E$ in the margin of the table header. For exploratory purposes these are probably accurate enough, but for final analyses, and for publication, precise cell-by-cell values may be needed. The command:

@

continues estimation of the current model until there is no discernible movement in the *fitted* values. The exact residuals can then be examined as above. The *Switches**Fit* option repeats this information, and allows you to choose your own magnitude[5] for 'discernible'.

---

2 If the table matches the full data then, and only then, the 'misclassified in that table' figure corresponds to the conventional 'overall percentage of cases misclassified by the model'.

3 Other programs watch the movement of the fitted values, and continue until these display only infinitesimal change.

4 Some simple models, though, will always be exactly fitted - the *Residual* menu will tell you.

5 By default, this command stops iteration when no fitted cell value increases or decreases by more than .004 (additive) at any 'correction' step.

## 3.3 Odds ratios on data and residuals

The *Data**Odds* option gives the 'interstitial' odds-ratios (reported as natural logarithms) for the observed values, on the table you specify; these can be read as one representation of the 'structure' of a table, being invariant under marginal transformations. The estimated[6] standard errors of the coefficients (on the observed values) are also reported. The *Residuals**Odds* option will also report the odds-ratios for the *fitted* values. Useful for understanding data (and models).

As an example of the display conventions, suppose, in *Residuals**Odds* that the repOrt reads in part:

```
LogOdds: Observed (with standard deviation) and {Fitted}
 ╔═══ A by B ═══
 ╟        ╔═══════B1╫═══════B2╫═══════B3╢
      A1  ╟         ╟        ╟        ╢
          ╟═══      .50      .30    ═══╢
          ╟     {  .40} {  .20}       ╢
      A2  ╟     (  .10) (  .15)       ╢
          ╚═══════════╩════════╩════════╝
```

then this says that, for the *observed* data, the logarithm of

the *ratio* of $(a_1b_1)/(a_1b_2)$ *to* $(a_2b_1)/(a_2b_2)$ *is* .5

The figure in round brackets gives the corresponding expected standard error (here .1). The logarithms of the odds-ratio on the *fitted* values are enclosed in 'curly' braces, thus '{}'. Any unreported logodds can be found by simple addition; thus for this table, on the *fitted* values

logodds $(a_1b_1)/(a_1b_3)$ to $(a_2b_1)/(a_2b_3)$ is given by .4+.2

*i.e.* in any *r*×*c* table, from the *(r-1)*×*(c-1)* set of 'interstitial' odds-ratios we can derive the entire set of odds ratios (which is what you would expect, remembering degrees of freedom).

As an aid to readily reading the import of the standard-error information, MicLog, through the *Switches**Odds* menu, can be set to display any chosen confidence range for the odds-ratios. You choose a confidence interval, and MicLog calculates the values lying at the appropriate fraction of the estimated standard error from the observed value. Thus, with a 95% confidence interval chosen, the above table would have displayed as:

```
LogOdds: <Observed; 95% range> and {Fitted}
 ╔═══ A by B ═══
 ╟        ╔═══════B1╫═══════B2╫═══════B3╢
      A1  ╟         ╟        ╟        ╢
          ╟═══  /  .30\ /  .01\   ═══╢
          ╟     {  .40} {  .20}       ╢
      A2  ╟     \  .70/ \  .59/       ╢
          ╚═══════════╩════════╩════════╝
```

which you could read as saying that, for example, the *population* logodds corresponding to the observed .5 has a 95% chance of falling in the range .3 to .7; this presents no *fresh* information, being simply a re-presentation[7] of the first table. As before, the {.4} refers to the *fitted* values, and these are omitted from the *Data**Odds* report. On colour monitors some attempt is made by the (flamboyant) use of hues to render the tables more legible.

Remember that the table can be specified at any required level of aggregation (so ABDE3 would display AB within D, for cases whose E value is 3, aggregating over C), though obviously category qualifiers would be inappropriate if applied to the first two variables in the

---

6 For the logarithmic 'odds' calculated on cells *a,b,c,d* the estimated standard error is given by: $\sqrt{(1/a + 1/b + 1/c + 1/d)}$

7 Thus: if *x* is the observed logodd, and *s* its estimated standard error, in the 95% case it reports values of '$x \pm 1.96s$'; and so on.

list.

## 4.0 TOPOLOGICAL VARIABLES

One of the novel features of MicLog is the ease with which the user can construct and deploy 'topological' constructed variables.

For the source references and an idiosyncratic review of the disutility of the technique see Macdonald (1983)[1]. A topological array allows the allocation of particular cells within a *k*-way table to exhaustive and exclusive categories, and these categories then define a set of 'marginals' which, as with the manifest marginals of the given variables, can be used in the definition of a model.

Using standard variables we can see ourselves as saying 'If we know the *Origin* distribution, and the *Destination* distribution, can we reproduce the data?'. We might further wish to say 'Suppose that additionally we know the number of *inheritors*, does our prediction improve?'. This would embody some substantive notion that a significant datum about a society is knowledge of the amount of 'inheritance', and if we have that, in addition to other marginal information, then we can adequately reproduce the observed complex *Origin* to *Destination* association.

*For example*. Suppose I have data on *Origin by Destination* (three categories each) *by Cohort*. I wish to construct an 'inheritors *vs* others' variable. I invoke the *Topology**Define* menu, specifying that I am concerned with topological variable *$A* (MicLog - its own convention this - identifies topological variables by dollar-sign and letter) and ask to *Input* a fresh topology. I then say I wish to define my topology over OD, and type the nine numbers in the OD array, thus:

```
1  2  2
2  1  2
2  2  1
```

differentiating *inheritors* ('1') from *others* ('2'). (MicLog prompts as it goes along, so you should not need to refer to this text.) I then estimate the model:

```
ON,DN,$A
```

That model, in addition to the aggregate [ON] table and the aggregate [DN] table, imports knowledge of the aggregate totals for 'inheritors' and 'others'. Return to the *Topology* menu will allow me to display the topology's parameters, to modify the topology, to re-estimate the model, and so on.

Another way of thinking about the operation, is that, having accepted the OD association as important, it would be helpful analytically to point to the features of the complex OD pattern which 'do the work'. If, in our example, *$A* were to do nearly as well as OD in the model then we might read it as indicating the central structure of OD, for this model. Topologies can help simplify in patterned ways, and so better understand, complex associations.

MicLog gives detailed prompts with the menus, so what follows section will indicate the facilities available, without always entering into the full detail of their invocation; when you sit at the keyboard all should become clear. The standard MicLog can deploy **7** active topologies, defined as *$A* to *$G*, and they can be saved on disk for later use.

### 4.1 Inclusion of topological variables

A topological variable, once defined, can be entered as a single variable into the model specification; it can be added and subtracted like any other variable.

---

1 'On the interpretation of a structural model of the mobility table' *Quality and Quantity* ***17***:203-224

For example

```
ON,$A,DN
-$A
+$B
```

would (provided *$A* and *$B* were defined) be a valid command sequence (and would at the last step estimate `[ON][DN][$A][$B]`)

MicLog will not however accept, in model input, terms composed of the *interaction* between topological variables and others (thus [N$A] is not, for MicLog, a valid term in a model). **But**, the desired effect can be exactly achieved by making a fresh topological variable defining this interaction, through the use of the *\T\D\A\eXtend* option (see *Section 4.5b*). This latter route has the advantage of making clear the location of each category effect.

## 4.2 Effect parameters from topological variables

Though in general MicLog (on good theoretical grounds) eschews effect parameters for interaction effects, it does provide them for topological variables, as some guide to their interpretation and modification. Coefficients are reported as natural logarithms, adjusted around the mean 'effect' of the variable. For some analyses (generally those with 'sparse' topologies) it may be useful to retain, throughout changes to the topology, the '*category appearing in most cells*' as the baseline; the *\Switches\Reference* menu provides for this.

In either presentation the coefficients are displayed when the model is estimated; they can also be seen neatly laid out in tabular form by invoking *\Topology\Show* for the topology concerned. [If the rank order of *coefficients* and *category numbers* do not match, the *\Topology\Order* option will be available to place the category numbers in effect order. But this is merely to help your thinking as you modify category allocations - the categories always operate purely *nominally* in the estimation.]

*Note* that *if* a model does not contain the marginals of the table on which an included topological variable was defined *then* the coefficients MicLog reports for that variable will not correspond to standard (*e.g.* GLIM) estimates - though the $G^2$ and the fitted values will of course be correct. MicLog could have been set to automatically insert marginals, but one might wish explicitly to create a model excluding them. Anyway, MicLog mutters, so you can insert the 'missing' marginals, and recompute, if you wish.

## 4.3 Input of topologies

The *\Topology\Define* menu initially allows you:

to *Copy* any other currently active topological variable

to *Get* a topology[2] from a file (which would have been written by *\T\D\Save*)

to *Input* from the keyboard.

To *Input* from the keyboard you first specify the table over which the topology will be defined (in our example above this would have been OD, but you can choose a three-way or more complex table if you wish). You then input the category allocations, in conventional sequence (MicLog prompts if you are in doubt). If at any point you type a negative integer all as-yet-unassigned cells are set to that (absolute) category.

*Hint* For 'sparse' topologies on large tables (most cells allocated to one category, a few cells differentiated) you may save time by first setting *all* cells to one level (type

---

2 obviously MicLog will 'get' only topologies originated on the current data (or on one with identical arrangement of variables and categories)

'-1' in response to the first category request) and then use *Topology**Change* to set the few specific cells.

The *Topology**Define**Title* option allows you to attach (and modify) a one-line title to the topology, as a note to self of your intentions. And topologies can be saved on disk (see *Section 4.7*).

## 4.4 Small modifications to topologies

*One cell*: The *Topology**Change* option allows you to change any specified cell of your defining table to a fresh category.

*Several cells*: The *Topology**Reinput* option will scan across each category of the defining table, reminding you of its current allocation, and inviting you to specify a new category; if you just press [Enter] the existing allocation is retained. (You could obviously achieve the same end result by several *Changes* or by an *Input*, but *Reinput* may save some work.)

*Hint:* If you are working with, say, topology *$B*, at the main prompt move straight to it, thus:

```
» t b
```

to avoid being told yet again what a topology is (see *Section 1.4* above).

## 4.5 Major modifications to topologies

Suppose we have data, ODNC, on *Origin by Destination by Nation by Cohort*, and we have defined some topology, *$A*, of *Origin by Destination*. Characteristic 'major' modifications might be:

*(a)* to wonder whether some detail of this topological pattern might vary for some one nation, or

*(b)* to wonder whether we might allow *effects* to vary across nations (whilst retaining the same topological *pattern* across all nations).

Obviously you could, tediously, simply start from scratch and redefine, using *Topology**Define**Input*, the topology. But the *Topology**Define****Additional*** menu provides less tedious, and less error-prone, routes. Let us consider the speculations in turn.

(a) *Vary the detail for some Nation* (say Nation 2). Having defined *$A* over OD only we cannot directly access *$A within* a Nation. But we can reset the Table on which we define *$A* to the three-way table ODN, using the *Table* option in this (*T**D**Additional*) menu. MicLog will now *display $A* across ODN. At this point the *definition* of *$A* remains unchanged; we see a set of identical OD topologies for each nation, and the *effect* of *$A* (within any model) remains unchanged. What has changed is our ability to access cells; the *Topology**Change* option will now let us, say, reset category 2 of O, 3 of D for Nation 2 to some new category, whilst leaving the pattern as it was for all other Nations.

For several modifications, to scan across the entirety of such an expanded table definition (with *Reinput*) could become tedious. So the *T**D**Additional* menu provides a *Within* option, which behaves exactly as *Reinput* (prompting and such) but allows you to select *some specified slice* of the entire defining table.

So you could 'reinput' categories *Within* some specified Nation (MicLog would *reinput* the entire OD table for that Nation), or you could operate *Within* some specified category of O and some specified category of D across nations (Miclog would *reinput* the specified OD cell for all Nations).

(b) *Same structure, different effects.* Let us return to the original *$A* defined over OD. If

we now use the *eXtend* option in the *T\D\Additional* menu, to 'extend' the definition over Nation, MicLog will reproduce the current *pattern* of *$A* using *different categories* for each Nation. So our earlier *inheritors*/*others* structure generates the following pattern:

```
1  2  2   for Nation 1      3  4  4   for Nation 2
2  1  2                     4  3  4
2  2  1                     4  4  3
```

and so on. This redefined *$A* (using more degrees of freedom) will obviously behave differently from the original *$A*; it maintains one aspect of the structure, but not the effect parameters, constant across countries. It can be modified, using *\T\Change* and *\T\D\A\Within*, as described under speculation *(a)*.

## 4.6 Data recodes and topologies

MicLog preserves topologies, if possible, across data recodes.

For example, returning to the data, ODNC, on *Origin by Destination by Nation by Cohort*, on which we defined some topology, *$A*, of *Origin by Destination*. Suppose we wish, for some analytic purpose, to collapse ODNC to an ODN table for analysis (disregarding cohort information). This would be achieved by invoking *\Data\Recode* and setting all categories of C to 1 to elide *Cohort* information (see *Section 6.1*). Since *$A* is still defined over this transformation it remains available. If, however, we now decide to combine categories 1 and 2 of *Destination*, then *$A* will remain defined if, *and only if*, these two columns of the topology (defined on *Origin by Destination*) are identical. MicLog keeps track and reports appropriately.

*Hint* If you wish to apply the same topology to data at different levels of aggregation, first define (and save) the topology using the fully disaggregated data, then recode the data as required.

## 4.7 Saving topologies on disk

A topology can be saved in a file by using the *\Topology\Define\Save* option. It can then be attached again to the data (or to another of identical structure[3]) by the *Get* option mentioned above (*Section 4.3*).

If you are only interested in *temporary* preservation of the current state of some topology before amending it, then perhaps exit from the *\Topology* menu, reenter specifying another topological variable, use the *Copy* option to place the contents of the first topology into this second topology, and work with the copy.

*Note* By default MicLog deletes all topologies from current memory (those *\Saved* on disk are unaffected) when fresh data are read. Such deletion could be an inconvenience if you wish to apply the same model to a clutch of identical datasets (say files carrying identical variables on differing countries); so it can be *suppressed* by the *\Switches\Topology* menu. *But note further*: you could equally *Save* and repeatedly *Get* the topologies; if you suppress deletion and read data of *different* structure the program may crash; and *\Data\Append* ***always*** deletes the existing topologies.

---

3 Note that if you have recoded categories (with *\Data\Recode*) the saved topology will only match the *recoded*, not the raw, data.

## 4.7 Mindless modification of topologies

Mindless improvement of $G^2$ is stupid; but, when constructing a topology you might be open to a minor modification which would noticeably increase $G^2$ provided that it did not subvert your theoretical concerns? Anyway, in the *Topology**Mindless* menu (which becomes available when you have estimated a model deploying the topological variable in question) MicLog provides two routes to suggested 'improvement' of the last-estimated model. Both involve considerable computation, so be prepared to exercise patience[4].

*Move*: computes MicLog's suggestion for the one-cell reallocation (retaining the original total number[5] of categories assigned to the topology) which most improves $G^2$.

*Add*: will report which allocation of a single cell to an additional new category would most increase $G^2$

MicLog will report the average improvement of competing solutions (so you can judge the pre-eminence of the one proffered). Under either strategy the choice is reported to you for inspection, and you are given the opportunity of incorporating it if you wish. The menu also provides an option to *Set* reporting detail (for the remainder of the session) so that you can be told the $G^2$ from *every* alternative considered - a more detailed check on how much 'better' the 'best' solution was.

## 4.8 Topologies find outliers?

The 'topology' provision can be used[6] to locate one variety of 'outlier'.

Consider the following. With data [ABCD] in place we have been considering the model 'ABC,CD'. Now we invoke the *Topology* menu, for variable *$G*, set the defining table to the full data table, [ABCD], and type '-1' as the first topology cell entry (this gives a featureless topology). We then estimate the model '$G,ABC,CD' (which obviously does not improve on the fit of 'ABC,CD'). Finally, we reenter the *Topology* menu, and activate the *Mindless**Add* option. After some (considerable) computation, MicLog will inform us of that cell which, if exactly fitted[7], would most improve the fit of the model. This may well differ from the cell least-well fitted (either in proportional or absolute measures) by the model. Food for thought.

---

4 The maths-coprocessor version (see *Appendix*) can help.

5 Before it starts, *Move* will renumber each category to the ordinal number of its effect in the last-estimated model, for we only considers 'moves' to *adjacent* (so defined) categories - experience has shown that this is as reliable as, but much faster than, searching 'all possible' moves.

6 I am indebted to Dr G Upton, University of Essex, for this suggestion.

7 A topology category assigned solely to a *single* cell in the full table provides information on the total of that (single) cell; and so fits it exactly.

# 5.0 DATA INPUT/OUTPUT

A central feature of MicLog's disk data format is its read/write compatibility with James A. Davis's program, CHIP. So MicLog data can be subjected to an alternative analysis strategy[1], and MicLog has access to an interesting and extensive range of data. MicLog can also read files from SPSSx 'Tables' output, files in various GLIM formats. All of these are read by the *Data**Get* option; the program itself detects the type of file encountered, and acts appropriately. It can also input data at the keyboard (by *Data**Input*).

The CHIP/MICLOG files are 'self describing', carrying variable and category labels. Variable and category labels can be attached to 'plain' files (SPSSx, GLIM, keyboard) by invoking the *Data**Save* option.

## 5.1 Input of MicLog or CHIP data

The *Data**Get* option will read, from disk, data prepared for CHIP or data saved by Miclog (by the *Data**Save* option). Such files are self-describing - they carry a title, names for each variable and for each category of each variable.

There are two minor differences between MicLog and CHIP filehandling (though each can read the other's files). Files from both programs carry full variable names, but a file with 'CHIP' origins will invite you, in MicLog, to select a set of single letters with which to refer to these variables within loglinear models (files saved by MicLog remember these single-letter assignments). Both programs allow you to specify a full filename (name *plus* extension); MicLog allows you to simply give a name, and supplies its own default extension (`MIC`).

Both programs allow you to precede a filename by a full directory address[2], so can readily access data anywhere on your installation.

## 5.2 Combining tables.

The *Data**Append* option allows tables to be combined into larger tables - for example it would allow three files, each carrying *Origin by Destination by Cohort* for a Nation, to be combined into one data table giving *Origin by Destination by Cohort by Nation*.

*Formally*: assume that you have read (say by *Data**Get*) an *n*-way table.

If the file referenced by *Data**Append* contains an:

*n-way table* it and the existent data are combined to a *(n+1)*-way table with two categories on the last variable.

*(n-1)-way table* then this file is combined as an additional category of the last variable of the existent data.

Sounds messy, but in practice straightforward (in both cases the tables are checked to be compatible in number of categories).

So, suppose NATION1, NATION2 and NATION3 contain the three subtables (say *Origin by Destination by Cohort* for three nations); then a *Data**Get* of NATION1, followed by a *Data**Append* of NATION2 and then a *Data**Append* of NATION3 will generate a *Origin*

---

1 See J. A. Davis (1987) *Social Differences in Contemporary America* (this book is associated with and describes the computer program CHIP, for analysing tables). The program, and its extensive associated American database, can be purchased separately from *True BASIC Inc*.

2 The default location for data files can be also reset from the *Switches**X**D* menu.

*by Destination by Cohort by Nation* table. The program guides you through the steps; *Data**Save* can place the final table on disk (for later one-stage input).

*Data**Append* (unlike *Data**Get*) will handle only MicLog/CHIP format files; if you need to manipulate say SPSSx Table files, then use a '*Data**Get* then *Data**Save*' sequence to convert to MicLog format, before invoking *Data**Append*. Mildly irritating, but not incapacitating.

## 5.3 Table input from keyboard

The *Data**Input* option will read data from the keyboard; you specify the number of variables[3], their identifying letters, and the number of categories in each. Tables are input with the first variable changing its categories most rapidly, the second next, and so on. Since MicLog prompts for each entry this should not be confusing, but perhaps practice with tiny tables before reading a mammoth one.

**Hint**: if confronted by a table in the following (fairly standard) layout):

| | | | S | |
|---|---|---|---|---|
| P | Q | R | 1 | 2 |
| 1 | 1 | 1 | a | b |
| | | 2 | c | d |
| 1 | 2 | 1 | e | f |

*etc* ...

when asked by Miclog to assign names to variables do so in the 'right-to-left' sequence S,R,Q,P (with S assigned to variable 1); MicLog will then prompt for data elements in the 'natural' sequence

```
a,b,c,d,e...
```

If you intend any serious analysis of the data you should now save them on disk (see *Section 5.6*) lest you need them later; the *Data**Save* command enables you also to attach labels to the *categories* of the variables, which may be helpful when examining detailed tabulations.

### 5.3.1 'Integer' or 'real' data

MicLog can handle integer (commonest) or real data. Calculations on 'real' data (numbers with decimal places) take about 25% longer than calculations on integer data: MicLog's preferred data mode is accordingly *integer*. When reading data, MicLog will opt for the data mode defined by the data itself (and tell you if it is *real*); the program also automatically moves to *real* data following any transformation which generates non-integer values (*e.g.* the *Data** transformation). The *Switches**Data* menu allows conversion between modes - its only intelligent use may be to round *reals* to *integers* if that makes sense for your analysis (setting *integers* to *reals* has no effect on the accuracy of results; it simply slows estimation). The *fitted* values are of course always treated as *reals*.

## 5.5 Input of 'GLIM' tables.

As far as MicLog is concerned these can be in one of two forms; each assumes that only one table is contained in any one file.

---

3 Currently a maximum of 7 - if you need more let me know.

(a) Each line is a cell value (real or integer) accompanied by location values, in 'free' or fixed format. Most such tables can be read in 'free' format (which requires that each number is separated from the next by a separator - characteristically one or more spaces). For example:

```
1066     1   1   1
1218.4   2   1   1
678      3   2   1     etc
```

This would place '678' in the cell identified by category 3 of A, category 2 of B, category 1 of C (MicLog allows you to change these default variable letters). MicLog recovers the number of categories in each variable by inspection of the file. Cells need not appear (though they usually do) in any particular sequence, and omitted cells are assigned the value zero. If MicLog encounters any real numbers, such as 1218.4, it sets its data mode to *real* (see *Section 5.3.1*)

(b) Each line carries one cell value, in determinate sequence.

```
1110
1126     etc
```

Here MicLog has to ask you for the number of variables and the number of categories in each variable, where the *first* variable is the one whose subscripts vary first, the second the variable whose subscripts vary next, and so on. Suppose we say three variables, A,B,C, each with two categories; then this assumes that the cell values are as if in the following sequence

```
          A   B   C
1110      1   1   1
1126      2   1   1
5678      1   2   1
1231      2   2   1
1231      1   1   2     etc
```

## 5.6 Saving the data on disk

A file saved by MicLog can be read and analysed by **CHIP** exactly as if it were a CHIP-originated file.

The *Data**Save* option will write the current data to a file of your choosing (if a file of that name already exists, MicLog checks that you intend to overwrite it before proceeding). The procedure also allows you to specify a one line descriptive 'title' for the file, and to attach 8-character labels to each variable, and to each category of each variable. These help render subsequent tables more legible (a reason to 'save' plain tables even if you intend never again to read them).

**Filenames** can be full DOS filenames, including disk and directory specification. If no disk and directory are specified, the current active directory is assumed[4]. If no extension is specified, the extension `.MIC` is supplied.

**Precision** *Integer* data is recorded exactly; if MicLog is in *real* mode (see *Section 5.3.1*) up to three decimal places are recorded.

---

[4] The default location for data files can be reset from the *Switches**X**D* menu.

# 6.0 DATA MANAGEMENT

The data modification and recode commands operate upon the copy of the data held within MicLog; the file copy is amended only if the data are subsequently saved over the original file.

## 6.1 Correcting a cell

The *Data**Cell* option allows you to set a particular[1] cell of the full data to some fresh value (which may be integer or real - see *Section 5.3.1*).

## 6.2 Recoding variables

The *Data**Recode* option allows you to recode variables, delete categories, or completely collapse variables (and so remove them from the analysis).

The program will prompt for the variable to be recoded, and then, for each existent category, ask for the new category assignment. If there are to be $k$ new categories, then the new category numbers you give must lie in the range 1 to $k$ inclusive (so no gaps); more than one existent category can be assigned to any new category.

For example, if R (region) has four categories then:

```
          Old/New
   N-Eng  1   3
   Wales  2   1
   S-Eng  3   3
Scotland  4   2
```

would ***combine*** original regions 1 and 3 into fresh region 3, place original 2 in 1, and original 4 in new 2. Miclog tries to behave intelligently about category labels. So, new regions 1 and 2 would carry the labels 'Wales' and 'Scotland'; category 3 would be assigned the provisional label '1+3', but you would be given the chance to supply a descriptive label (such as 'England').

Not all 'old' categories need be allocated an active 'new' location. By specifying *zero* for the new category number, we **remove** that category from the data.

For example, if we further recode our recoded Region variable, thus:

```
          Old/New
   Wales  1   1
Scotland  2   0
 England  3   2
```

this will remove the Scots from further analysis, reducing total cases, leaving us with only regions '1' and '3' (now renumbered '1' and '2', but retaining their appropriate labels).

If *all* of the retained categories of a variable are recoded to the value 1 then that variable will carry no discriminating evidence (all its categories are identical) and it is therefore elided from subsequent analyses. In other words, we can turn an $n$-way tabulation into an (n-1)-way tabulation. This elision itself does not affect number of cases, but it can be combined with deletion(s).

For example, consider a combination of elision and deletion. If we had recoded our three category Region variable thus:

---

[1] if your problem is that you missed out an entry in keyboard input (from *Data**Input*), then write the file to disk (*Data**Save*) and use your favourite editor or word-processor to insert the number in its appropriate sequence amongst the data - echo the layout that you see.

```
          Old/New
   Wales  1   1
Scotland  2   0
 England  3   1
```

then this would have had *two* effects: the Scots are removed *and* R (Region) ceases to be a variable; so if our original full data were say [ABCRY] we now have an [ABCY] table (with slightly fewer cases, since the Scots have been deleted).

Recodes are cumulative - if you want to return to the original data, reread them by using *Data**Get*. Note also that, as described in *Section 4.6*, the program does its best to preserve any active topologies as recoding proceeds.

## 6.3 Capture of fitted values

MicLog can replace the *observed* data by the *fitted* values from a model. This can have some pedagogic uses - if we input data, fit some specified model, capture and save the fitted values we then have a data set which exactly matches some known model, and others can then be sent in search of it. It is also of some use as a general exploratory tool (and in 'mostellerising' - see below). The command:

**!**

replaces the data[2] by the fitted values from the last-estimated model. Since fitted values will not be exact integers, the data mode is automatically switched to *real* (see *Section 5.3.1*); you might possibly wish to 'round' this new data to the nearest integers, by the *Switches**Data menu,* to more closely simulate 'ordinary' data.

## 6.4 Changing category size

The *Data** option allows you to multiply a particular category of a particular variable by some constant: one way to explore counterfactuals about your data distribution.

Suppose, for example, categories 2 and 3 of C have sums x and y respectively; then two invocations of \Data* , one to multiply C2 by (x+y)/2x and one to multiply C3 by (x+y)/2y should give equal numbers in categories 2 and 3 of C without disturbing overall sample size. Whether such possible transformations are sensible transformations is for you to decide (see Section 7.1 for a more sweeping strategy).

Note that a *non-integer multiplier* will yield non-integer data, so in that case the data mode is automatically switched to *real* (see *Section 5.3.1*) to retain full precision. (You can, if you wish, 'round' this new data to the nearest integers, by the *Switches**Data menu,* to retain 'ordinary'-seeming data.)

---

[2] The original data file is of course unaffected: if we wish to retain the new values we must write the data to file, by *Data**Save*.

# 7.0 MISCELLANEOUS

Some facilities of questionable utility:

## 7.1 Equalising marginals ('Mostellerising')

Some feel that removing variations in table marginals allows us more clearly to 'see' the structure of the data. That suggestion can be seen as a modification of the iteration strategy involved in loglinear modelling:

In standard modelling, when you specify marginals, MicLog starts with an estimated data array of exactly equal cells, and imposes upon it the pattern of the observed marginals you specify.

In *Mostellerising* mode (*Switches* menu), when you specify marginals, MicLog starts with the *observed* data array, treats mention of marginals as specifying *notional tables with equal*[1] *cells*, and imposes these constraints on the data.

Thus, in Mostellerising mode, given data [ABCD], specifying the model as A,B would give, as the fitted values, the data adjusted to be equally spread across categories of A and across categories of B. If we specify AB we get as fitted values the data constrained to equal spread across the categories of [AB]; and so on. If you then replace the actual data by the constrained data, using the ! command (see Section 6.3), you can inspect or analyse these counterfactual data.

For example, in Mostellerising mode (the main prompt changes to 'M»' as a reminder) if we had simple *Origin by Destination* (OD) data, then the request:

```
O,D
```

would generate a table, retaining the interaction structure of the original, but with equal marginals within each of O and D. Or again, given *Origin by Destination data within Nations and Cohorts* `[ODNC]`, the sequence (in Mostellerising mode):

```
[N]
!
```

would give us, for analysis, data adjusted to have equal numbers in each Nation.

## 7.2 Handling Marginal zeroes in tables

Data whose fitted terms involve marginal zeroes have been held to present a problem. Cells which add to these fitted zeroes must themselves be zero (and so exactly[2] fitted). The exact fitting of cells summing to a zero marginal has been held to accord an improper primacy to such cells.

One suggested 'solution' (but see Upton 1986) is to add a small constant to each cell of the data (thus making the assumption that the zeroes represent 'low frequency', not zero, entries). The *Switches**Zero* menu allows Miclog to see the data 'as if' a small constant were added to each cell of the observed data, and subsequent models will then be estimated on this notional data.

Note that the data themselves are not modified by this switch (so recode and data modification commands are unaffected by it, likewise the *Data**Save* command).

---

1 The ability to stipulate specified, non-uniform, target marginals might be more useful: that should appear in the next release of MicLog.

2 This implied exact fitting disturbs the standard formula for the degrees of freedom; MicLog calculates and reports the modified value (and notes 'marginal zeroes') in those circumstances.