Loglinear Analysis MT 1.0

for GAUSS™ Mathematical and Statistical System

Aptech Systems, Inc.

Part Number: `004436`
Version 1.0
Documentation Revision: 1861   June 7, 2010

Contents

Installation 1

1.1 UNIX/Linux/Mac

If you are unfamiliar with UNIX/Linux/Mac, see your system administrator or system documentation for information on the system commands referred to below.

1.1.1 Download

1. Copy the `.tar.gz` or `.zip` file to `/tmp`.
2. If the file has a `.tar.gz` extension, unzip it using `gunzip`. Otherwise skip to step 3.

   `gunzip` app_*appname*_*vernum*`.`*revnum*_`UNIX.tar.gz`

3. `cd` to your **GAUSS** or **GAUSS Engine** installation directory. We are assuming `/usr/local/gauss` in this case.

   ```
   cd /usr/local/gauss
   ```

4. Use `tar` or `unzip`, depending on the file name extension, to extract the file.

   ```
   tar xvf /tmp/app_appname_vernum.revnum_UNIX.tar
   ```
   – or –
   ```
   unzip /tmp/app_appname_vernum.revnum_UNIX.zip
   ```

1.1.2 CD

1. Insert the Apps CD into your machine's CD-ROM drive.
2. Open a terminal window.
3. `cd` to your current **GAUSS** or **GAUSS Engine** installation directory. We are assuming `/usr/local/gauss` in this case.

   ```
   cd /usr/local/gauss
   ```
4. Use `tar` or `unzip`, depending on the file name extensions, to extract the files found on the CD. For example:

   ```
   tar xvf /cdrom/apps/app_appname_vernum.revnum_UNIX.tar
   ```
   – or –
   ```
   unzip /cdrom/apps/app_appname_vernum.revnum_UNIX.zip
   ```
   However, note that the paths may be different on your machine.

1.2 Windows

1.2.1 Download

Unzip the `.zip` file into your **GAUSS** or **GAUSS Engine** installation directory.

1.2.2 CD

1. Insert the Apps CD into your machine's CD-ROM drive.

2. Unzip the `.zip` files found on the CD to your **GAUSS** or **GAUSS Engine** installation directory.

1.2.3 64-Bit Windows

If you have both the 64-bit version of **GAUSS** and the 32-bit Companion Edition installed on your machine, you need to install any **GAUSS** applications you own in both **GAUSS** installation directories.

1.3 Difference Between the UNIX and Windows Versions

- If the functions can be controlled during execution by entering keystrokes from the keyboard, it may be necessary to press ENTER after the keystroke in the UNIX version.

Getting Started 2

2.1 Getting Started

GAUSS 6.0.26+ is required to use these routines. See **`_rtl_ver`** in `src/gauss.dec`.

2.1.1 Setup

In order to use the procedures in the **Loglinear Analysis MT** module, the **LLMT** library must be active. This is done by including **`loglinmt`** in the **`library`** statement at the top of your program or command file:

```
library loglinmt,quantal,pgraph;
```

This enables **GAUSS** to find the **LLMT** procedures. An **llControl** structure is used to hold control variables that are used by the procedures in this module. To define an instance of this structure and set its members to default values, include the following instructions just after the **library** statement at the top of your program:

```
#include loglinmt.sdf
struct llControl llc;
llc = llControlCreate;
```

The version number of each module is stored in a global variable. For **Loglinear Analysis MT**, this global variable is:

_llmt_ver 3×1 matrix. The first element contains the major version number, the second element the minor version number, and the third element the revision number.

If you call for technical support, you may be asked for the version of your copy of this module.

2.1.2 README Files

If it exists, the file `README.llmt` contains any last minute information on the **Loglinear Analysis MT** procedures. Please read it before using them.

Loglinear Analysis MT 3

The **Loglinear Analysis MT** module contains a set of routines for the analysis of categorical data using loglinear analysis. These routines can estimate most of the models considered in such texts as Y.M.M. Bishop, S.E. Fienberg and P.W. Holland (1975) *Discrete Multivariate Analysis*, A. Agresti (1984) *Analysis of Ordinal Categorical Data*, and S. Haberman (1979) *Analysis of Qualitative Data*, Volumes 1 and 2.

Estimation is performed in this module with the procedure **`llest`** which computes maximum likelihood estimates by the Newton-Raphson method. The other routines are for entering data, constructing design matrices, easily passing information to **`llest`**, and printing the results. Input tables can be entered with the **GAUSS** editor, from the keyboard using the program **`ltable`**, or with a table constructed from raw data using the **GAUSS** procedure **`crosstab`**.

3.1 The Loglinear Model

This section presents the basic ideas related to the loglinear model. This is done primarily to clarify the notation that will be used below. The reader who is not familiar with these models should consult one of the texts given in the references.

Consider a K-way table constructed from variables $V_i(i = 1, K)$, with variable V_i having K_i categories. The resulting table contains $N = K_1 * K_2 * \ldots K_K$ cells. The observed variables for this table are contained in the $N \times 1$ vector n. Let the N cell indices associated with n be contained in the $K \times N$ matrix T where element t_{ij} is the category level of variable V_j for n_i. For example, for a $2 \times 2 \times 2$ table ($K = 3$):

	A=1			A=2	
	C=1	C=2		C=1	C=2
B=1	1	2	B=1	5	6
B=2	3	4	B=2	7	8

n and T would be:

$$n = \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \end{matrix} \qquad T = \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 2 \\ 1 & 2 & 1 \\ 1 & 2 & 2 \\ 2 & 1 & 1 \\ 2 & 1 & 2 \\ 2 & 2 & 1 \\ 2 & 2 & 2 \end{matrix}$$

Let m be an $N \times 1$ vector of expected cell frequencies for the loglinear model. The loglinear model is defined as:

$$\log m = D\beta$$

for an $N \times P$ design matrix D with P parameters in the $P \times 1$ vector β.

Estimation is based on the assumption that the n_i are independent Poisson random variables with expected values equal to m_i. β is estimated by the Newton-Raphson method, based on an algorithm presented in Agresti (1984).

The loglinear model is overparameterized, which means that constraints must be imposed on the parameters. Different programs use different constraints. Most commonly, parameters are constrained to sum to zero over the level of an individual variable. Alternatively, the parameter associated with the first level of any variable can be fixed to zero. **`llest`** allows either set of constraints.

The specific loglinear model is defined by its design matrix. The **Loglinear Analysis MT** module contains a number of procedures for automatically constructing common kinds of design matrices. Alternatively, the user can construct his or her own design matrix to pass to **`llest`**.

An important class of models is called hierarchical models. These models are uniquely specified by the configurations that are fit by the model. A configuration is defined as a table formed from the original table by collapsing over dimensions of the table. For example, V_1V_3 would be the table formed from $V_1V_2 \ldots V_K$ by collapsing over V_2, V_4, . . ., V_K. For purposes of describing hierarchical models in the procedures described below, variables will be referred to by number (e.g., 1, 2) and configurations will be defined by integers containing the variables in the table (e.g., the configuration for variables 1 and 3 is indicated as 13).

Tables with cell weights can also be analyzed. Thus, the model being estimated is somewhat more general than indicated above. Specifically, the model with weights is:

$$\log(m_i/z_i) = \beta_k X_{ik}$$

where z_i is the cell weight associated with cell i. See Haberman (1979) for details on this form of the model. Clogg and Eliason (1988) provide a number of examples of using cell weights.

In its simplest form, weighting allows the analysis of incomplete tables. Incomplete tables are tables in which some of the cells are to be excluded from analysis. These cells are considered to have structural zeros. To indicate which cells have structural zeros, an $N \times 1$ vector Z is constructed with a 1 for cells to be analyzed and a 0 for cells to be dropped from analysis. The example file `llratemt.e` provides an example of weighting.

3.2 General Design

The **Loglinear Analysis MT** module consists of a set of main procedures, auxiliary procedures called by the main procedures, and an **`llControl`** structure containing control variables which specify aspects of estimation and output. The main procedures are:

Procedure	*Description*	*Page*
`letters`	Set the **`lettrs`** control variable to a string array of labels.	4-1
`ll3way`	Estimate all 3-way hierarchical models for a table.	4-2
`lldesign`	Input a design matrix for use with **`llest`**.	4-3
`llest`	Compute maximum likelihood estimates for a loglinear model.	4-4
`llhier`	Estimate a hierarchical model given its fit configurations.	4-5
`llout`	Print the estimates for a loglinear model.	4-6
`lltable`	Input a table for use with **`llest`**.	4-8
`lluser`	Estimate a model with a user-supplied design.	4-9

`llest` is the central routine that estimates a loglinear model given a specific table, a set of weights, and a design matrix. **`llest`** will make efforts to fix an improper design matrix. For example, columns with all zeros are deleted and a warning is printed. Further, if singularities are encountered that cannot be eliminated, the offending design matrix is printed along with an explanation. The degrees of freedom for the model are computed with methods suggested by Clogg and Eliason (1988). Essentially, the rank of the design matrix is subtracted from the number of cells without either structural or fitted zeros.

To see how **`llest`** is used, consider the hierarchical model 12,3 for the $2 \times 2 \times 2$ table 123. Assume that all cells are to be analyzed. The loglinear model could be estimated as:

```
library loglinmt;
#include loglinmt.sdf

struct llControl llc;
llc = llControlCreate;
n = { 32, 86, 11, 35, 6, 73, 41, 70 };
d = { 1   1   1   1   1   1,
      1   1   1  -1   1  -1,
      1   1  -1   1  -1   1,
      1   1  -1  -1  -1  -1,
      1  -1   1   1  -1  -1,
      1  -1   1  -1  -1   1,
      1  -1  -1   1   1  -1,
      1  -1  -1  -1   1   1 };

{ m,b,v,gsq,df,tol } = llest(llc,n,0,d);
```

llest returns various useful pieces of information that are discussed below. However, the procedure **llout** translates this information into a convenient form. Thus, readable results could be obtained by adding the matrix of cell indices and calling **llout**:

```
       t = { 1  1  1,
             1  1  2,
             1  2  1,
             1  2  2,
             2  1  1,
             2  1  2,
             2  2  1,
             2  2  2 };
llout(llc,t,m,b,vc,gsq,df,tol,n,0,d,0,0,0);
```

In general, it is tedious and error-prone to enter design matrices and tables. The procedure **ltable** prompts the user for the number of variables, the number of categories per variable, and the value for each cell. The user is given the option of specifying cell weights. The user can provide a name (e.g., “myname”) which the program will use to place the table indices, cell counts, and weights in global matrices (here, *mynamet*, *mynamen* and *mynamez*). If desired, these matrices will be saved to disk.

The procedure **`lldesign`** prompts the user for the number of rows and columns in the design matrix and prompts for each element of the design matrix. The rows correspond to cells in the table and the columns correspond to parameters. The user can provide a name (e.g., "myname") which the program will use to place the design matrix in a global matrix (here, *mynamed*). If desired, the matrix will be saved to disk.

If the design matrix, table, and cell indices have been entered, the procedure **`lluser`** can be used to pass the table indices, cell counts and user-defined design matrix to **`llest`**, and print the results with **`llout`**. For example:

```
library loglinmt;
#include loglinmt.sdf

struct llControl llc;
llc = llControlCreate;
loadm myn, myt, myd;
call lluser(llc,myt,myn,0,myd,"","");
```

Having complete control over the design matrix is useful for many specialized models. Consider the row effects model presented on page 87 of Agresti (1985). Assume that the cell counts and table indices are saved in *agr87t* and *agr87n*. Then the row effects model could be estimated as:

```
/* 1 */  /* lluser.e:  Agresti, page 87 */
         library loglinmt;
         #include loglinmt.sdf

         struct llControl llc;
         llc = llControlCreate;
/* 2 */  loadm agr87t,agr87n;
/* 3 */  agr87d =
            { 1  1  0  1  0 -1  0,
                  1  1  0  0  1  0  0,
                  1  1  0 -1 -1  1  0,
                  1  0  1  1  0  0 -1,
                  1  0  1  0  1  0  0,
                  1  0  1 -1 -1  0  1,
                  1 -1 -1  1  0  1  1,
                  1 -1 -1  0  1  0  0,
                  1 -1 -1 -1 -1 -1 -1 };
/* 4 */  save agr87d;
/* 5 */  cls;
/* 6 */  output file = "llusermt.out" reset;
/* 7 */  print "llusermt.e:  Row Effects Model from Agresti, page 87.";
           print;
/* 8 */  call lluser(llc,agr87t,agr87n,0,agr87d,"","");
/* 9 */  output off;
```

The numbers to the left are for reference purposes only.

Line (1) labels the program.

Line (2) loads the cell frequencies and indices.

Line (3) inputs the design matrix.

Line (4) saves the design matrix for later use.

Line (5) clears the screen.

Line (6) creates the output file.

Line (7) labels the output.

Line (8) calls **llest** and **llout** and thus estimates the model.

Line (9) turns off the output.

This program is saved as `llusermt.e` in the `examples` subdirectory.

While manually entering the design matrix provides a great deal of flexibility, it is tedious and error-prone. For hierarchical models, there are two procedures that do most of the work:

ll3way computes all possible hierarchical models for a three-way table. All the user must provide is the cell indices and the table. For example, assuming that *llc*, *t*, *n*, and *z* are in memory:

```
ll3way(llc,t,n,z);
```

would estimate all models for the table with indices *t*, counts *n*, weights contained in *z*, and an instance of an **llControl** structure *llc*.

llhier estimates the hierarchical model described by the configurations that are to be fit. For example,

```
/* llhiermt1.e:  Fienberg, page 27. */
library loglinmt;
#include loglinmt.sdf

struct llControl llc;
llc = llControlCreate;
loadm fien27t,fien27n;
print;
output file = "llhiermt1.out" reset;
print "llhiermt1.e:  Fienberg, page 27.";
print;
print "Model 1:  Conditional independence.";
print;
cfg = { 13, 12 };
nm = "Fien3"$|"Fien2"$|"Fien1";
call llhier(fien27t,fien27n,0,cfg,nm);
print;
print "Model 2:  Complete independence.";
print;
cfg = { 1, 2, 3 };
call llhier(llc,fien27t,fien27n,0,cfg,nm);
output off;
```

would estimate two hierarchical models for the table stored in *fien27t* and *fien27n*. This program is saved as `llhiermt1.e` in the `examples` subdirectory.

letters sets the **lettrs** member of an **llControl** structure to a string array of labels. For example:

```
library loglinmt;
#include loglinmt.sdf

struct llControl llc;
llc = llControlCreate;
llc = letters(llc,"S X U V W")
```

would set *llc*.**llettrs** to a 5×1 string array with S, X, U, V and W in it. With **letters** it is possible to specify models with letters rather than numbers. For example: SU, VW rather than 13, 45.

The procedures **ll3way**, **llhier** and **lluser** illustrate the ways in which **llest** and **llout** can be used to construct customized applications for loglinear modeling.

llhier constructs a design matrix for hierarchical models, passes it to **llest** for estimation, and passes the results to **llout** for printing.

ll3way loops through all hierarchical models for three-way tables, passes the design matrix to **llest**, and prints the goodness of fit of each model.

lluser takes information on a table and design matrix and passes them to **llest**, and then passes the results to **llout**.

Taking these three procedures as models, one could easily create a specialized procedure for logit type models, models for ordered data, and so on.

3.3 Control Variables

The members of the `llControl` structure are control variables, which are used in **Loglinear Analysis MT** to specify various options. Chapter **??** provides details on which members of this structure affect which procedures. The defaults can be changed by modifying the `llControlCreate` procedure in `llapplmt.src`. Alternatively, the defaults can be changed by setting new values immediately before using the **Loglinear Analysis MT** procedures.

`lettrs` $K \times 1$ string array, contains the characters that will be used as mnemonics for labeling the output. The first element is associated with the first variable, the second with the second, etc. The abbreviations supplied in **`lettrs`** are used to label the output in **`llout`**. By default, A, B, ... are used as abbreviations for variables.

`maxit` scalar, indicates the maximum number of iterations. If convergence is not reached within 25 iterations, it generally means that there is a problem with the model. When **`maxit`** is exceeded, a warning is printed along with the results. These results are to help you assess the problem; they are NOT correct. Default = 25.

`mult` scalar. Values are:

0 print coefficients for the loglinear model.

1 print coefficients for the multiplicative model. The parameters in the multiplicative models are simply the exponential of the parameters in the loglinear model.

Default = 0.

`prtd` scalar. If 1, print the design matrix. Default = 1.

`prte` scalar. If 0, the estimates of the parameters are not printed. This is useful if you just want the chi-squares or expected cell frequencies. Default = 1.

`prtit` scalar. If 1, print the information on iterations. Default = 1.

prtt scalar. If 1, print the table of cell counts, expected values, residuals and weights. Default = 1.

sum0 scalar. Values are:

0	parameters for first level of any variable are constrained to zero.
1	parameters in the hierarchical model are constrained to sum to 0 across the level of any variable.

Default = 1.

tol scalar. Sets tolerance for convergence. Tolerance is defined as the maximum difference between the coefficients of the current iteration and the previous one. Default = 1e−5.

two scalar. Values are:

0	do not double or square the coefficients. This is useful if logit models are being estimated.
1	print two times the additive coefficients or the squared multiplicative coefficients.

Default = 0.

3.4 References

1. Agresti, A. 1984. *Analysis of Ordinal Categorical Data.* New York:Wiley.
2. Bishop, Y.M.M, S.E. Fienberg and P.W. Holland (1975) *Discrete Multivariate Analysis*. Boston:MIT Press.
3. Clogg, C.C. and S.R. Eliason. 1988. “Some Common Problems in Log-Linear Analysis.” Pp. 226-257 in Long, *Common Problems/Proper Solutions*. Beverly Hills:Sage University Press.
4. Fienberg, S.E. 1980. *The Analysis of Cross-Classified Categorical Data*. Boston:MIT Press.

Loglinear Analysis MT Reference 4

letters

PURPOSE Sets the **lettrs** member of an **llControl** structure to a $K \times 1$ string array of labels.

LIBRARY **loglinmt**

FORMAT *llc2* **= letters(***llc1***,***str***);**

INPUT	*llc1*	an instance of an **llControl** structure
	str	string, containing a list of one character variables
OUTPUT	*llc2*	a copy of *llc1* with the **lettrs** member set according to the input in *str*:

`llc2.`**`lettrs`**	$K \times 1$ string array, contains the characters that will be used as mnemonics for labeling the output. The first element is associated with the first variable, the second with the second, etc. The abbreviations supplied in **`lettrs`** are used to label the output in **`llout`**. By default, A, B, ... are used as abbreviations for variables.

REMARKS For example:

```
llc = letters(llc,"S X U V W")
```

would set **`llc.lettrs`** to a 5×1 string array with S, X, U, V and W in it. With **`letters`**, it is possible to specify models with letters rather than numbers. For example: SU, VW rather than 13, 45.

EXAMPLE

```
library loglinmt;
#include loglinmt.sdf

struct llControl llc;
llc = llControlCreate;
loadm fien27t,fien27n;
llc = letters(llc,"S D H");
cfg = "SH"$|"SD";
nm = "SPECIES"$|"DIAMETER"$|"HEIGHT";
call llhier(llc,fien27t,fien27n,0,cfg,nm);
```

SOURCE `llapplmt.src`

SEE ALSO **`llhier`**

ll3way

PURPOSE	Estimates all hierarchical models for a three-way table.
LIBRARY	**loglinmt**
FORMAT	**ll3way(***llc***,***t***,***n***,***z***);**

INPUT *llc* an instance of an **llControl** structure. The following members of **llc** are referenced within the **ll3way** routine:

llc.**header** string, specifies the format for the output header. **llc.header** can contain zero or more of the following characters:

t	print title (see **llc.title**)
l	bracket title with lines
d	print date and time
v	print procedure name and version number

Example:

```
llc.header = "tld";
```

If **llc.header** == “”, no header is printed. Default = “tldv”.

llc.**maxit** scalar, indicates the maximum number of iterations. If convergence is not reached within 25 iterations, it generally means that there is a problem with the model. When **llc.maxit** is exceeded, a warning is printed along with the results. These

	results are to help you assess the problem; they are NOT correct. Default = 25.
`llc.output`	scalar. If 1, intermediate results are printed. Default = 1.
`llc.prtit`	scalar. If 1, print information on iterations. For this procedure, it is best to set **`llc.prtit`** to 0. Default = 1.
`lrc.ranktol`	scalar, the tolerance used to determine if any of the singular values are effectively 0 when computing the rank of a matrix. Default = 10^{-13}.
`llc.title`	string, message printed at the top of the results. Default = “”.
`llc.tol`	scalar. Sets tolerance for convergence. Tolerance is defined as the maximum difference between the coefficients of the current iteration and the previous one. Default = 10^{-5}.

t	$N \times P$ matrix of cell indices, where N equals `rows(n)` and P equals the number of rows in the coefficient matrix.
n	$N \times 1$ matrix of cell counts of table.
z	$N \times 1$ matrix with cell weights.

OUTPUT None.

REMARKS Likelihood ratio and Pearson chi-squares for all seventeen possible loglinear models for three-way tables are printed.

The procedures **`_llhdsgn`**, **`_lldigit`** and **`_llsubcn`** are called by this procedure. These are documented in the file `llapplmt.src`.

EXAMPLE

```
library loglinmt;
#include loglinmt.sdf

struct llControl llc;
llc = llControlCreate;
loadm fien27t,fien27n;
llc.prtit = 0;
print "ll3waymt.e:  See Fienberg, page 42.";
ll3way(llc,fien27t,fien27n,0);
```

SOURCE `llapplmt.src`

SEE ALSO **`llest`**, **`ltable`**, **`lldesign`**, **`llout`**, **`llhier`**, **`lluser`**

lldesign

PURPOSE Interactively input a design matrix for analysis with **`llest`**.

LIBRARY **`loglinmt`**

FORMAT **`lldesign;`**

INPUT The user is prompted for the number of rows and columns in the design matrix. The rows correspond to the cells of the table; the columns to the parameters to be estimated. The user is then prompted to enter elements of the design matrix. Finally, the user is asked to specify a name for the matrix.

OUTPUT None.

REMARKS Assume that *name* is the user-supplied name to specify the matrices. **`lldesign`** then creates the global design matrix *name*`d`.

The user is given the option to save the outputs to disk.

EXAMPLE

```
library loglinmt;
#include loglinmt.sdf

struct llControl llc;
llc = llControlCreate;
lltable;
/* respond to prompts from lltable to create myt, myz and myn */
lldesign;
/* respond to prompts from lldesign to create myd */
call lluser(llc,myt,myn,myz,myd,"","");
```

SOURCE `llinputmt.src`

SEE ALSO **`llest`**, **`ltable`**, **`llout`**, **`llhier`**, **`ll3way`**, **`lluser`**

llest

PURPOSE Estimates a loglinear model by maximum likelihood.

LIBRARY **`loglinmt`**

FORMAT { *m*,*b*,*vc*,*gsq*,*df*,*tol* } = **`llest(`***llc*,*n*,*z*,*d***`);`**

INPUT *llc* an instance of an **`llControl`** structure. The following members of **`llc`** are referenced within the **`llest`** routine:

`llc.`**`maxit`**	scalar, indicates the maximum number of iterations. If convergence is not reached within 25 iterations, it generally means that there is a

	problem with the model. When `llc.maxit` is exceeded, a warning is printed along with the results. These results are to help you assess the problem; they are NOT correct. Default = 25.
`llc.prtit`	scalar. If 1, print information on iterations. Default = 1.
`lrc.ranktol`	scalar, the tolerance used to determine if any of the singular values are effectively 0 when computing the rank of a matrix. Default = 10^{-13}.
`llc.tol`	scalar. Sets tolerance for convergence. Tolerance is defined as the maximum difference between the coefficients of the current iteration and the previous one. Default = 10^{-5}.

	n	$N \times 1$ matrix of cell counts of the table, where N is the number of cells.
	z	$N \times 1$ matrix with cell weights.
	d	$N \times P$ design matrix, where P is the number of parameters to be estimated. The rows of *d* correspond to the cells in *n*.
OUTPUT	*m*	$N \times 1$ vector of expected cell frequencies, in the same order as *n*.
	b	$P \times 1$ vector of parameter estimates.
	vc	$P \times P$ covariance matrix of *b*.
	gsq	scalar, likelihood ratio chi-square.
	df	scalar, degrees of freedom associated with the model. The degrees of freedom are computed as the difference between the number of nonzero fitted cells and the rank of the design matrix.

tol	scalar, the actual tolerance at convergence.

REMARKS Maximum likelihood estimation is performed by the Newton-Raphson method. The program is based on a SAS program written by Agresti (1984).

SOURCE `llestmt.src`

SEE ALSO **`ltable`**, **`lldesign`**, **`llout`**, **`llhier`**, **`ll3way`**, **`lluser`**

llhier

PURPOSE Estimates a loglinear hierarchical model defined by a set of configurations to be fit.

LIBRARY **`loglinmt`**

FORMAT { *m*,*b*,*covb*,*gsq*,*df*,*tol*,*dsgn* } = **llhier(***llc*,*t*,*n*,*z*,*cfg*,*nm***);**

INPUT *llc* an instance of an **`llControl`** structure. The following members of *llc* are referenced within the **`llhier`** routine:

`llc.`**`header`** string, specifies the format for the output header. **`llc.header`** can contain zero or more of the following characters:

t	print title (see **`llc.title`**)
l	bracket title with lines
d	print date and time
v	print procedure name and version number

	Example: `llc.header = "tld";` If **llc.header** == “”, no header is printed. Default = “tldv”.
`llc.lettrs`	$K \times 1$ string array, contains the characters that will be used as mnemonics for labeling the output. The first element is associated with the first variable, the second with the second, etc. The abbreviations supplied in **llc.lettrs** are used to label the output in **llout**. By default, A, B, ... are used as abbreviations for variables.
`llc.maxit`	scalar, indicates the maximum number of iterations. If convergence is not reached within 25 iterations, it generally means that there is a problem with the model. When **llc.maxit** is exceeded, a warning is printed along with the results. These results are to help you assess the problem; they are NOT correct. Default = 25.
`llc.mult`	scalar. Values are: **0** print coefficients for the loglinear model. **1** print coefficients for the multiplicative model. The parameters in the multiplicative models are simply the exponential of the parameters in the loglinear model. Default = 0.

`llc.output`	scalar. If 1, intermediate results are printed. Default = 1.
`llc.prtd`	scalar. If 1, print the design matrix. Default = 1.
`llc.prte`	scalar. If 0, the estimates of the parameters are not printed. Default = 1.
`llc.prtt`	scalar. If 1, print the table of cell counts, expected values, residuals and weights. Default = 1.
`lrc.ranktol`	scalar, the tolerance used to determine if any of the singular values are effectively 0 when computing the rank of a matrix. Default = 10^{-13}.
`llc.sum0`	scalar. Values are: **0** parameters for first level of any variable are constrained to zero. **1** parameters in the hierarchical model are constrained to sum to 0 across the level of any variable. Default = 1.
`llc.title`	string, message printed at the top of the results. Default = “”.
`llc.tol`	scalar, sets tolerance for convergence. Tolerance is defined as the maximum difference between the coefficients of the current iteration and the previous one. Default = 10^{-5}.
`llc.two`	scalar. Values are: **0** do not double or square the coefficients. This is useful if logit models are being estimated.

		1 print two times the additive coefficients or the squared multiplicative coefficients. Default = 0.
	t	$N \times P$ matrix of cell indices.
	n	$N \times 1$ matrix of cell counts of table.
	z	$N \times 1$ matrix of cell weights.
	cfg	$P \times 1$ string array of character labels of the – or – $P \times 1$ numeric vector of variable indices configurations to be fit.
	nm	$Q \times 1$ string array of variable names, or “” for no names.
OUTPUT	*m*	$N \times 1$ vector of expected cell frequencies, in the same order as *n*.
	b	$P \times 1$ vector of parameter estimates.
	covb	$P \times P$ covariance matrix of *b*.
	gsq	scalar, likelihood ratio chi-square.
	df	scalar, degrees of freedom associated with the model. The degrees of freedom are computed as the difference between the number of nonzero fitted cells and the rank of the design matrix.
	tol	scalar, the actual tolerance at convergence.
	dsgn	scalar, the design matrix that was estimated.

REMARKS Results of the estimation are also sent to the output device.

The procedures **`_llhdsgn`**, **`_lldigit`**, and **`_llsubcn`** are called by this procedure. These are documented in the file `llapplmt.src` .

EXAMPLE

```
library loglinmt;
#include loglinmt.sdf
```

```
struct llControl llc;
llc = llControlCreate;
cfg = { 13, 12 };
nm = "VAR_A"$|"VAR_B"$|"VAR_C";
call llhier(llc,t,n,0,cfg,nm);
```

SOURCE `llapplmt.src`

SEE ALSO **`ltable`**, **`lldesign`**, **`llout`**, **`llhier`**, **`ll3way`**, **`lluser`**

llout

PURPOSE Prints the results obtained from **`llest`**.

LIBRARY **`loglinmt`**

FORMAT **`llout(`***llc*`,`*t*`,`*m*`,`*b*`,`*vc*`,`*gsq*`,`*df*`,`*tol*`,`*n*`,`*z*`,`*d*`,`*xcfg*`,`*nm*`,`*lbl***`);`**

INPUT *llc* an instance of an **`llControl`** structure. The following members of **`llc`** are referenced within the **`llout`** routine:

`llc.`**`header`** string, specifies the format for the output header. **`llc.header`** can contain zero or more of the following characters:

t	print title (see **`llc.title`**)
l	bracket title with lines
d	print date and time
v	print procedure name and version number

Example:

```
llc.header = "tld";
```

If `llc.header` == “”, no header is printed. Default = “tldv”.

`llc.mult`	scalar. Values are: **0** print coefficients for the loglinear model. **1** print coefficients for the multiplicative model. The parameters in the multiplicative models are simply the exponential of the parameters in the loglinear model. Default = 0.
`llc.prtd`	scalar. If 1, print the design matrix. Default = 1.
`llc.prte`	scalar. If 0, the estimates of the parameters are not printed. Default = 1.
`llc.prtt`	scalar. If 1, print the table of cell counts, expected values, residuals and weights. Default = 1.
`llc.title`	string, message printed at the top of the results. Default = “”.
`llc.two`	scalar. Values are: **0** do not double or square the coefficients. This is useful if logit models are being estimated. **1** print two times the additive coefficients or the squared multiplicative coefficients. Default = 0.

t $N \times P$ matrix of cell indices.

m	$N \times 1$ vector of expected cell counts.
b	$P \times 1$ vector of parameter estimates.
vc	$P \times P$ covariance matrix of estimates in *b*.
gsq	scalar, likelihood ratio chi-square.
df	scalar, degrees of freedom associated with the model. The degrees of freedom are computed as the difference between the number of nonzero fitted cells and the rank of the design matrix.
tol	scalar, the actual tolerance at convergence. If this is not less than **`llc`**.**`tol`**, then the estimates did not converge.
n	$N \times 1$ vector of cell counts of table.
z	$N \times 1$ vector of cell weights.
d	$N \times P$ design matrix, where *P* is the number of parameters to be estimated. The rows of *d* correspond to the cells in *n*.
xcfg	scalar, configurations fit in hierarchical model. If the model is not hierarchical or you do not want the configurations printed, set *xcfg* = 0.
nm	$N \times 1$ string array of variable names, or “” for no names.
lbl	$P \times 1$ string array of labels to be associated with parameters being estimated. If “”, parameters will not be labeled.

OUTPUT Results are sent to the output device.

REMARKS This routine may be used to output the results from any use of **`llest`**.

EXAMPLE This example illustrates how **`llest`** and **`llout`** can be combined to create routines to simplify the estimation of loglinear models. This procedure is documented as **`lluser`**.

```
proc (0) = lluser(llc,t,n,z,d,nm,lbl);
   local m,b,vc,gsq,df,tol,xcfg;
```

```
      xcfg = 0;
      { m,b,vc,gsq,df,tol } = llest(llc,n,z,d);
      llout(llc,t,m,b,vc,gsq,df,tol,n,z,d,xcfg,nm,lbl);
   endp; /* lluser */
```

SOURCE `llestmt.src`

SEE ALSO **`llest`**, **`ltable`**, **`lldesign`**, **`llhier`**, **`ll3way`**, **`lluser`**

llControlCreate

PURPOSE Sets the members of an **`llControl`** structure to default values.

LIBRARY **`loglinmt`**

FORMAT *llc* **`= llControlCreate;`**

INPUT None.

OUTPUT *llc*	an instance of an **`llControl`** structure with its members set to default values

REMARKS Putting this instruction at the top of all programs that invoke **Loglinear Analysis MT** procedures is good practice. This will prevent control variables from being inappropriately defined when a program is run either several times or after another program that also calls **Loglinear Analysis MT** procedures.

SOURCE `llapplmt.src`

lltable

PURPOSE	Interactively constructs a table for analysis by **llest**.
LIBRARY	**loglinmt**
FORMAT	**lltable;**
INPUT	The user is prompted for the number of variables and the number of categories per variable. The user is then prompted to enter cell counts. Optionally, the user may specify cell weights. Finally, the user is asked to specify a name for the matrices.
OUTPUT	Assume that *name* is the user-supplied name to specify the matrices. **ltable** creates the following global matrices:

namen	$N \times 1$ matrix of cell counts of table, where N is the number of cells.
namez	$N \times 1$ matrix of cell weights.
namet	$N \times P$ matrix of cell indices associated with *namen*.

REMARKS	The user is given the option to save the outputs to disk.
SOURCE	`llinputmt.src`
SEE ALSO	**llest**, **lldesign**, **llout**, **llhier**, **ll3way**, **lluser**

lluser

PURPOSE	Estimates and prints results for a loglinear model based on a user-supplied design matrix.
LIBRARY	**loglinmt**
FORMAT	{ *m*,*b*,*covb*,*gsq*,*df*,*tol* } = **lluser(***llc*,*t*,*n*,*z*,*d*,*nm*,*lbl***);**

INPUT *llc* an instance of an **llControl** structure. The following members of **llc** are referenced within the **lluser** routine:

llc.**header** string, specifies the format for the output header. **llc.header** can contain zero or more of the following characters:

t	print title (see **llc.title**)
l	bracket title with lines
d	print date and time
v	print procedure name and version number

Example:

```
llc.header = "tld";
```

If **llc.header** == “”, no header is printed. Default = “tldv”.

llc.**maxit** scalar, indicates the maximum number of iterations. If convergence is not reached within 25 iterations, it generally means that there is a problem with the model. When **llc.maxit** is exceeded, a warning is

	printed along with the results. These results are to help you assess the problem; they are NOT correct. Default = 25.
`llc.mult`	scalar. Values are: **0** print coefficients for the loglinear model. **1** print coefficients for the multiplicative model. The parameters in the multiplicative models are simply the exponential of the parameters in the loglinear model. Default = 0.
`llc.output`	scalar. If 1, results are printed to the screen. Default = 1.
`llc.prtd`	scalar. If 1, print the design matrix. Default = 1.
`llc.prtit`	scalar. If 1, print the information on iterations. Default = 1.
`llc.prtt`	scalar. If 1, print the table of cell counts, expected values, residuals and weights. Default = 1.
`lrc.ranktol`	scalar, the tolerance used to determine if any of the singular values are effectively 0 when computing the rank of a matrix. Default = 10^{-13}.
`llc.title`	string, message printed at the top of the results. Default = “”.
`llc.tol`	scalar, sets tolerance for convergence. Tolerance is defined as the maximum difference between the coefficients of the current iteration and the previous

		one. Default = 10^{-5}.
	`llc.two`	scalar. Values are: **0** do not double or square the coefficients. This is useful if logit models are being estimated. **1** print two times the additive coefficients or the squared multiplicative coefficients. Default = 0.
	t	$N \times P$ matrix of cell indices.
	n	$N \times 1$ matrix of cell counts of table.
	z	$N \times 1$ matrix of cell weights.
	d	$N \times P$ design matrix, where P is the number of parameters to be estimated. The rows of *d* correspond to the cells in *n*.
	nm	$N \times 1$ string array of variable names, or “” for no names.
	lbl	$P \times 1$ string array of parameter names. If “”, parameters will not be given names.
OUTPUT	*m*	$N \times 1$ vector of expected cell frequencies, in same order as *n*.
	b	$P \times 1$ vector of parameter estimates.
	covb	$P \times P$ covariance matrix of *b*.
	gsq	scalar, likelihood ratio chi-square.
	df	scalar, degrees of freedom associated with the model. The degrees of freedom are computed as the difference between the number of nonzero fitted cells and the rank of the design matrix.
	tol	scalar, the actual tolerance at convergence.

REMARKS Results of the estimation are also sent to the output device if `llc.output` = 1.

EXAMPLE This example will estimate the row effect model for ordinal data.

```
library loglinmt;
#include loglinmt.sdf

struct llControl llc;
llc = llControlCreate;
loadm agr87t,agr87n;
agr87d =
  { 1  1  0  1  0 -1  0,
    1  1  0  0  1  0  0,
    1  1  0 -1 -1  1  0,
    1  0  1  1  0  0 -1,
    1  0  1  0  1  0  0,
    1  0  1 -1 -1  0  1,
    1 -1 -1  1  0  1  1,
    1 -1 -1  0  1  0  0,
    1 -1 -1 -1 -1 -1 -1 };
output file = llmt1.out reset;
call lluser(llc,agr87t,agr87n,0,agr87d,"","");
output off;
```

SOURCE `llapplmt.src`

SEE ALSO **`ltable`**, **`lldesign`**, **`llout`**, **`llhier`**, **`ll3way`**

Index