Linear Regression MT

for GAUSS™

Version 1.0

Aptech Systems, Inc.

Documentation Revision: 1502 March 8, 2010

Part Number: 004435

Contents

Chapter 1

Installation

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.

1.3 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.

Chapter 2

Linear Regression MT

2.1 Getting Started

The *LINEAR REGRESSION MT* or **LRMT** module is a set of procedures for the estimation of single equation and simultaneous equation models. Single equation models are estimated using *Ordinary Least Squares*. Systems of equations can be estimated using *Two-Stage Least Squares*, *Three-Stage Least Squares*, or *Seemingly Unrelated Regression*.

The core of this module consists of the following procedures:

l2sls	Linear Two-Stage Least Squares Regression
l3sls	Linear Three-Stage Least Squares Regression
lreg	Linear Regression by Ordinary Least Squares
lsur	Linear Seemingly Unrelated Regression

In addition to these estimation procedures, a procedure **lrtest** is provided for linear hypothesis testing of any of the above regression models.

Special features of the **LRMT** module include:

- Handles arbitrarily large data sets with multiple variables.
- Performs multiple linear hypothesis testing easily.
- Estimates regressions with linear restrictions.
- All regression procedures may operate on a specified range of observations.

- Performs iteratively re-weighted *Three-Stage Least Squares* and *Seemingly Unrelated Regression*.

This chapter begins with some general aspects of the use of the **LRMT** module. The second chapter provides additional topics covering the application of the procedures. Comprehensive details of each estimation procedure are provided in the last chapter.

2.1.1 README Files

The file `README.lrmt` contains any last minute information on this module. Please read it carefully before using the procedures in this module.

2.1.2 Version Number

The version number is stored in a global variable **_lrmt_ver**, which is a 3×1 matrix containing the major version, minor version, and revision number in that order.

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

2.2 Setup

There are four essential parts to any estimation procedure in this module. These must be specified in any programs that call these estimation procedures.

1. **Header:**

   The header consists of five statements: a **library** statement which activates the LRMT library, an **#include** statement to include the file containing the definitions of the structures used by **LRMT**, declarations of an instance of an **lrControl** structure and an **lrOut** structure, and a call to **lrControlCreate** which sets the members of the **lrControl** structure to default values. These five statements are specified at the top of the command file and should look something like this:

```
library lrmt, pgraph;
#include lrmt.sdf
struct lrControl lrc;
struct lrOut lro;
lrc = lrControlCreate;
```

In the example above, the PGRAPH library is necessary if you intend to use the Publication Quality Graphics.

2. **Data Setup:**

   Next, the user must specify the data to be passed to the procedures. For example, the format for **lreg** is:

   ```
   lro = lreg(lrc,dataset,dv,iv,restrict);
   ```

   Here is an example of the data setup for that procedure call:

   ```
   dataset = "translog";  // File name of the data set
   dv = "y1";            // Specify dependent variable
   iv = "const"$|"x1"$|"x2";  // Specify independent variables
   ```

3. **Specify Options:**

   Options are controlled by setting the corresponding members of the **lrControl** structure. Following the above example, you may want to analyze the data with both *influence* and *collinearity* diagnostics. This can be accomplished by specifying the following two statements:

   ```
   lrc.lregres = "residual";
   lrc.lregcol = 1;
   ```

4. **Calling the Procedure:**

   Each estimation procedure can print results to the screen and send output to the specified output file and/or return a global output structure to memory. If all you need is the printed results, you can call the procedure as follows:

   ```
   call lreg(lrc,dataset,dv,iv,0);
   ```

   In this case, you would not need to declare an instance of an **lrOut** structure at the top of the program.

   However, if you want information returned to memory, you must assign the result to an **lrOut** structure.

   ```
   lro = lreg(lrc,dataset,dv,iv,0);
   ```

   The resulting structure, *lro*, stores all of the return statistics in an efficient manner. The members of the **lrOut** structure that are relevant to each procedure are listed in the reference section.

2.3 Data Sets

A **GAUSS** data set is a binary disk file, which is saved with a `.dat` extension. The `.dat` file is comprised of the data and a header which contains the names and types of the variables associated with each column of the data set.

2.3.1 Data Transformations

It is assumed that the data set for analysis is ready before you call the procedures. If you need to modify your data, **GAUSS Data Tool** is available for fast and simple manipulation of data sets. **GAUSS Data Tool** provides users with a powerful and flexible environment for viewing and modifying data. It includes commands for keeping and dropping variables, selecting observations, sorting, merging on a key variable or set of variables, imputing missing data, and transforming data using **GAUSS** functions.

2.3.2 Creating Data Sets

There are three ways to create a **GAUSS** data set.

1. If you have an ASCII format data file, use the **ATOG** utility to convert it into a **GAUSS** data set. For details, see **ATOG** in the *UTILITIES* section of the **GAUSS** manual.
2. If you have a matrix in memory, use the command **create** or **saved** to create a data set. See the *COMMAND REFERENCE* section of the **GAUSS** manual.
3. **GAUSS Data Tool** has commands for creating new data sets and translating ASCII and Excel files to **GAUSS** data sets.

To look at a data set in **GAUSS**, use the keyword **datalist**. The syntax is:

```
datalist filename [variables];
```

For details, see **datalist** in the **GAUSS** manual.

2.4 Compiling the Procedures

By compiling your procedures and saving the compiled code to disk, you can eliminate the time required to compile the **LRMT** procedures into memory. The compiled file saved to disk will have a `.gcg` extension.

To create a file containing the compiled images of the procedures you use together often, you may, for example, type the following commands from the command line:

```
new;
library lrmt;
external proc lreg, l2sls;
saveall procset1;
```

The procedures listed in the **external** statement will be compiled and the compiled images will be saved to the file `procset1.gcg`. The file containing the compiled image should be saved on a subdirectory listed in the SRC_PATH of the **GAUSS** configuration file.

To use these procedures, you need to have the statement

```
    use procset1;
```

at the top of your command file. The **use** command will look along the SRC_PATH for the file you specify. A **library** statement may not be necessary if you are using only procedures that are saved in the file specified in the **use** statement.

2.5 Troubleshooting

Here are common error messages that you may encounter when using **LRMT** procedures.

```
  Undefined symbols:
      lrControlCreate    c:gauss\examples\test1.e(6)
      l3sls              c:gauss\examples\test1.e(22)
      lrtest             c:gauss\examples\test1.e(23)
         .
         .
         .
```

If this happens, the LRMT library may not be active. Check if the following statement is listed at the top of your command file:

```
   library lrmt;
```

2.6 Error Codes

When certain errors are encountered in the specification of the model or the data being analyzed, the procedures either terminate with an error message or return an error code in the **errcode** member of the **lrOut** structure. This is controlled with the low order bit of the trap flag. See **trap** in the **GAUSS** *COMMAND REFERENCE*.

TRAP 0	terminate with error message
TRAP 1	return scalar error code

2.6.1 Testing for Error Codes

The returning error code appears as a missing value if printed. Use **scalerr** to retrieve the error number.

```
    trap 1;        // Initialize the trap
    lro = lreg(lrc,dataset,dv,iv,restrict);
    if scalerr(lro.errcode);
       print "Error " scalerr(lro.errcode) " was encountered.";
       end;
    endif;
    trap 0;         // Reset the trap
```

Use **lrError** to display the error message assocated with an error code.

```
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Program file: testerrmt1.e
Data file:    tmt11_3
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/

library lrmt;
struct lrControl lrc;
struct lrOut lro1,lro2;
lrc = lrControlCreate;

dataset = "tmt11_3";
output file = testerrmt1.out reset;
trap 1;
dv = "y";
iv = "const"$|"p1"$|"p3";
restrict = "p1 + p33 = 1";       // User mistypes p3
lro1 = lreg(lrc,dataset,dv,iv,restrict);
if scalerr(lro1.errcode);
   lrerror("Error located in the 1st equation",lro1.errcode);
   pause(3);
endif;

dv = "y";
iv = "const"$|"p1"$|"p4";  // Variable p4 is not in the data set
lro2 = lreg(lrc,dataset,dv,iv,0);
if scalerr(lro2.errcode);
   lrerror("Error located in the 2nd equation",lro2.errcode);
   pause(3);
endif;
trap 0;
output off;
```

NOTE: The example files are included in the `examples` subdirectory.

2.6.2 List of Error Codes

Following is a list of error code definitions:

1	Data file not found.
2	Variables specified not found in the data set.
4	Invalid range.
5	Type mismatch.
21	Misspecification in the restriction string.
22	The restricted equations are inconsistent.
23	The restricted equations are linearly dependent.
24	Singular values not all computed.
30	System singular.
31	There are fewer observations than parameters to estimate.
36	Variables specified are not consistent.
40	The input **lrOut** structure contains an error.
74	The file for residual diagnostics cannot be opened.
75	There is not enough disk space to write the residual measures.
76	Unable to find eigenvalues.

2.7 Getting Help on Procedures

All of the procedures in the **LRMT** module are automatically accessible in **GAUSS** if the LRMT library is active. You can find the definition of an **LRMT** procedure and information about its syntax and arguments as follows:

If you are running Windows, place the cursor on the name of the procedure and press Ctrl-F1.

If you are running UNIX, type browse followed by the name of the procedure.

Chapter 3

Topics in Linear Regression MT

This chapter covers a wide variety of application examples in **LRMT**, from the estimation of single equations to systems of equations. The examples cover the following topics:

1. Tests for heteroskedasticity
2. Test of structural change
3. Estimating a translog cost function using *Ordinary Least Squares*
4. Estimating a system of cost share equations using of *Seemingly Unrelated Regression*
5. Estimating Klein's Model I using *Three-Stage Least Squares*

In order to run some of the examples below, the **dataloop** translator must be turned on. If you are running Windows, select "Translate Dataloop Cmds" from the **Run** menu to turn the **dataloop** translator on. If you are running UNIX, use the **config** command.

3.1 Tests for Heteroskedasticity

Heteroskedasticity exists when the errors do not have a constant variance. Its consequences lead to inefficient least squares estimators and biased estimator of the variances. Any inferences based on these estimates could be misleading. There are several tests which can detect the existence of heteroskedasticity. Two of them are discussed below.

3.1.1 The Breusch-Pagan Test

This is a Lagrange Multiplier test and covers a wide range of heteroskedastic situations. It assumes the model disturbances are distributed normally with variance as follows:

$$\sigma_i^2 = \sigma^2 f(\alpha^0 + Z_i'\alpha)$$

where f is any unspecified functional form. Z_i is a vector of variables which you suspect influence the heteroskedasticity, and α is a vector of coefficients. If $\alpha = 0$, the model is said to be homoskedastic.

Procedures for this test are given as below:

1. Run **lreg** and obtain both the residual vector (*lro1*.`res`) and the residual sum of squares (*lro1*.`sse`).
2. Set *sse* equal to *lro1*.`sse`, and calculate the $\tilde{\sigma}^2$ as follows:

$$\tilde{\sigma}^2 = \frac{sse}{n}$$

3. Rerun **lreg** with the form as below and obtain *lro2*.`sst` and *lro2*.`sse`.

$$\frac{\hat{e}_t^2}{\tilde{\sigma}^2} = \alpha_0 + Z_t'\alpha + V_t$$

   where $\hat{e}_t$ are the least squares residuals from step 1 and $\alpha_0 = 1$.

4. Set *sst* equal to *lro2*.`sst` and *sse* equal to *lro2*.`sse` and then compute the test statistic, which is

$$LM = (sst - sse)/2$$

   where *sst* and *sse* are respectively the total sum of squares and residual sum of squares obtained from step 3.

Under the null hypothesis, the test statistic is asymptotically distributed as Chi-squared with degrees of freedom equal to the number of regressors (k) in Z. Thus, at 5% level if $LM > \chi^2_{0.95}(k)$, you reject the hypothesis of homoskedasticity.

■ Example

In the example below, X_2 is thought to be the influential variable. With the use of **lreg**, you must specify a file name to hold the residual vector and its diagnostic measures. This can be done by assigning a file name to the **lrControl** structure member **regres** (i.e., lrc.lregres = "temp"). Do not confuse $\tilde{\sigma}^2$ and $\hat{\sigma}^2$. $\tilde{\sigma}^2$ is calculated in step 3 and uses n as divisor. $\hat{\sigma}^2$ is one of the return statistics, namely **s2**, stored in the output structure, and it uses $(n - k)$ as divisor.

```
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Program file: hetermt1.e
Data set:     hetermt1
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/

library lrmt;
#include lrmt.sdf

struct lrControl lrc;
struct lrOut lro1,lro2;
lrc = lrControlCreate;

dataset = "hetermt1";
outset = "hetermt1.out";
output file = ^outset reset;
dv = "y";
iv = "const"$|"x2"$|"x3";

lrc.output = 0;
lrc.lregres = "temp";          /* Perform the 1st regression */
lro1 = lreg(lrc,dataset,dv,iv,0);
sse = lro1.sse;
n = lro1.nobs;
newS2 = sse/n;

dataloop temp newdatamt;      /* Calculate the new dependent */
   extern newS2;            /* Variable as step 3 */
   make newDV = (res^2)/newS2;
   keep newDV x2;
endata;

dataset = "newdatamt";
dv = "newDV";
iv = "const"$|"x2";
lro2 = lreg(lrc,dataset,dv,iv,0);  /* Perform the 2nd regression */
sse = lro2.sse;
sst = lro2.sst;
chisq = (sst-sse)/2;         /* Compute the test statistic */
format /rd 12,4;
print "Total sum of squares:    " sst;
print "Residual sum of squares: " sse;
print "Chi-Squared statistic:   " chisq;

output off;
```

```
/*+++++  end of program file  ++++++++++++++++++++++++++++++++++*/
```

Here is the output:

```
Total sum of squares:          52.8982
Residual sum of squares:       43.6927
Chi-Squared statistic:          4.6027
```

By comparing the $\chi^2_{(1)}$ value at 5% significance level, which is 3.84, you may conclude that heteroskedasticity exists in the disturbance variances.

3.1.2 The Goldfeld-Quandt Test

The central idea of this test is to split the observations into two groups. And under the null hypothesis of homoskedasticity, both groups should have equal variance. Whereas under the alternative, the disturbance variances would not be the same. In this test, observations are sorted according to the magnitude of the independent variable X_i, and this variable is hypothesized to be related to the variance of disturbances. Goldfeld and Quandt suggest that a certain number of the middle observations be omitted to increase the distinction between the error variances.

The test procedures are as follows:

1. Sort the observations according to the values of X_i, where X_i is thought to be the influential variable.

2. Drop some central observations, the number (c) to be dropped is very subjective and is not obvious.

3. Run two separate regressions, on the first and last $(n - c)/2$ observations, and find out their corresponding residual sums of squares.

4. Compute the test statistic as follows:

$$R = \frac{sse_2}{sse_1}$$

   where sse_1 and sse_2 are respectively the residual sums of squares from the first and second regressions. In other words, $sse_1 = lro1$.`sse` and $sse_2 = lro2$.`sse`.

Under the null hypothesis, the test statistic R is distributed as F with $[(n - c - 2k)/2, (n - c - 2k)/2]$ degrees of freedom. If $F > F_{0.95}$, the homoskedasticity is rejected at 5 percent level.

■ Example

The data used in this example is per capita expenditure on public schools and per capita income by state in 1979. Data is from the United States Department of Commerce (1979, p.157). Since the Goldfeld-Quandt test requires the data to be ordered, **sortd** is used to sort the data set with the income variable as the sorting key. Total number of observations is 51. Each regression is run with 17 observations. By assigning a data range to the global variable *lrc*.`range`, you run the regression with the indicated range.

```
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Program file: hetermt2.e
Data set:     hetermt2
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/

library lrmt;
#include lrmt.sdf

struct lrControl lrc;
struct lrOut lro1,lro2;
lrc = lrControlCreate;

outset = "hetermt2.out";
output file = ^outset reset;

  /* Sort the data according to the values of income */
sortd("hetermt2","newdatamt","income",1);

lrc.output = 0;
dv = "expense";
iv = "const"$|"income";

lrc.range = { 1,17 };            /* 1st regression with the */
lro1 = lreg(lrc,"newdatamt",dv,iv,0);  /* first 17 observations */
sse1 = lro1.sse;

lrc.range = { 35,51 };           /* 2nd regression with the */
lro2 = lreg(lrc,"newdatamt",dv,iv,0); /* last 17 observations */
sse2 = lro2.sse;

format /rd 12,6;
print "SSE from the 1st regression:   " sse1;
print "SSE from the 2nd regression:   " sse2;
print "The F-statistic for this test: " sse2/sse1;

output off;
```

```
/*+++++  end of program file  ++++++++++++++++++++++++++++++++++*/
```

Here is the output:

```
SSE from the 1st regression:   28809.327473
SSE from the 2nd regression:   86642.410198
The F-statistic for this test:     3.007443
```

Since at 5% level of significance $F > F_{0.95}(15, 15)$, where $F_{0.95} = 2.4$, you would reject the hypothesis of homoskedasticity.

3.2 Test of Structural Change

The *Chow* test is one of several ways to test the differences in parameter estimates across data sets. Suppose you have two data sets:

$$Y_i = X_i\beta_i + \varepsilon_i \qquad i = 1, 2$$

where Y_i has n_1 observations, Y_2 has n_2 observations, and X_1 and X_2 both have k regressors k. In matrix notation, the two regressions can be expressed as follows:

$$Y = \left[\begin{array}{c} Y_1 \\ Y_2 \end{array}\right] = \left[\begin{array}{cc} X_1 & 0 \\ 0 & X_2 \end{array}\right]\left[\begin{array}{c} \beta_1 \\ \beta_2 \end{array}\right] + \left[\begin{array}{c} \varepsilon_1 \\ \varepsilon_2 \end{array}\right] = X\beta + \varepsilon \tag{1}$$

Equation (1) is the unrestricted form of the model. Its residual sum of squares can be obtained from the two separate regressions (i.e., $e'e = e_1'e_1 + e_2'e_2$).

To test whether $\beta_1 = \beta_2$, we specify the restricted model:

$$Y = \left[\begin{array}{c} X_1 \\ X_2 \end{array}\right]\beta_1 + \varepsilon = X^*\beta_1 + \varepsilon \tag{2}$$

The test statistic for the null hypothesis is an F statistic and is defined as follows:

$$F = \frac{(e_*'e_* - e'e)/k}{e'e/(n_1 + n_2 - 2k)}$$

where $e'_* e_*$ and $e'e$ are respectively the restricted and unrestricted residual sums of squares, n_1 is the number of observations in the first sample, n_2 is the number of observations in the second sample, and k is the number of regressors.

Under the null hypothesis, if $F > F_{0.95}(k, n_1 + n_2 - 2k)$, you would reject the hypothesis at the 5% level that the coefficient vectors are the same in two samples.

Example

This example is from Maddala [12, page 131]. The data set, `chow.dat`, presents data on per capita food consumption, price of food and per capita income for the years 1927-1941 and 1948-1962. We wish to test the stability of the parameters in the demand function between the two periods. The estimated function is as follows:

$$\ln q = \alpha + \beta_1 \ln P + \beta_2 \ln Y$$

where q is the food consumption per capita, P is the food price, and Y is the consumer income.

Since the data needs to be in logged, **dataloop** is used to transform the data. Three regressions are run with the desired range of data in order to generate their corresponding residual sums of squares. Finally the test statistic is calculated. Note that the **lrControl** structure member **range** is used to control the data range to be passed into the regressions.

```
/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Program file: chowmt.e
Data set:     chowmt
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/

library lrmt;
#include lrmt.sdf

struct lrControl lrc;
struct lrOut lrot,lro1,lro2;
lrc = lrControlCreate;

dataloop chowmt newdatamt;
   consume = ln(consume);   /* Take natural log of variables */
   price = ln(price);
   income = ln(income);
endata;

output file = chowmt.out reset;
dv = "consume";
iv = "const"$|"price"$|"income";
lrc.output = 0;
lrot = lreg(lrc,"newdatamt",dv,iv,0);    /* Full sample run */
sseR = lrot.sse;

lrc.range = { 1,15 };    /* Run with the first 15 observations */
lro1 = lreg(lrc,"newdatamt",dv,iv,0);
sse1 = lro1.sse;
n1 = lro1.nobs;

lrc.range = { 16,30 };   /* Run with the last 15 observations */
lro2 = lreg(lrc,"newdatamt",dv,iv,0);
sse2 = lro2.sse;
n2 = lro2.nobs;

sseU = sse1 + sse2;    /* Calculate the unrestricted sse */
F = ((sseR - sseU)/3)/(sseU/(n1+n2-2*3));
prob = cdffc(f,3,(n1+n2-2*3));

format /rd 12,8;
print "unrestricted residual sum of squares: " sseU;
print "  restricted residual sum of squares: " sseR;
print "                         F statistic: " F;
print "                  significance level: " prob;
output off;
```

```
/******  end of program file  +++++++++++++++++++++++++++++++++++*/
```

Here is the output:

```
unrestricted residual sum of squares:   0.00169541
  restricted residual sum of squares:   0.00286947
                        F statistic:   5.53995446
                 significance level:   0.00491253
```

The restricted and unrestricted residual sums of squares are different from those calculated in Maddala [12, page 131]. This is due to differing presentation of the results. On page 113, Maddala uses $10^2 \times$SSE to present both residual sums of squares. From the F-tables, $F_{0.95}(3, 24) = 3.01$ and $F_{0.99}(3, 24) = 4.72$. Thus, even at the 1% level of significance, the hypothesis of stability is rejected.

3.3 Estimating a Translog Cost Function Using Ordinary Least Squares

The duality of cost and production functions is an important subject in neoclassical economics. According to the duality theory, under appropriate regularity conditions, all of the information about the solution to the production function can be obtained via the corresponding cost function. In fact, Silberberg [13] has suggested that the duality theory assures us that if a cost function satisfies some elementary properties, i.e., linear homogeneity and concavity in the factor prices, then there is also a unique production function. Homogeneity in input prices implies that when all input prices are doubled, the cost of production also doubles (i.e., mathematically, $C(tP, Y) = t \cdot C(P, Y)$ where $t > 0$, C is the cost and is a function of the input prices (P) and output (Y)).

Although the topic of estimation of the cost function is very broad, some of the interesting points are presented below. This section demonstrates the practical aspects of estimating a translog cost function with symmetry and homogeneity in input prices imposed. The usage of the translog functional form is due to its popularity and flexibility.

The translog cost function is specified as below:

$$\ln C = \alpha_0 + \alpha_y \ln Y + \sum_i \alpha_i \ln P_i + \frac{1}{2}\beta_{yy}(\ln Y)^2$$
$$+\frac{1}{2}\sum_i \sum_j \beta_{ij} \cdot \ln P_i \cdot \ln P_j + \sum_i \gamma_{yi} \cdot \ln Y \cdot \ln P_i$$

where C is the total cost, Y is the level of output, and P_i is the i^{th} input price. If symmetry is assumed (i.e., $\alpha_{ij} = \alpha_{ji} \;\; \forall \;\; i \neq j$), fewer parameters are estimated.

Several hypotheses can be tested within the function. They are constant returns to size (or linear homogeneity in output, i.e., the same idea as linear homogeneity in input prices) and functional form of the cost function (i.e., the cobb douglas technology). The conditions for constant returns to size are tested by restricting: $\alpha_y = 1$, $\alpha_{yy} = 0$, and $\alpha_{yi} = 0 \;\; \forall \;\; i$. And the hypothesis regarding the Cobb Douglas technology is tested by restricting all quadratic terms and cross product terms to be zero. Both hypotheses require an F-test and the test statistic is defined in the **lrtest** procedure.

Finally, in order to guarantee that the cost function is homogenous of degree one in input prices, some restrictions must be imposed into the function. For the translog case, it is: $\sum_i \alpha_i = 1$, $\sum_i \gamma_{yi} = 0$, and $\sum_j \beta_{ij} = 0 \;\; \forall \;\; i$. However, if the Cobb Douglas technology cannot be rejected, the following restrictions are required: $\sum_i \alpha_i = 1$, all quadratic terms and cross product terms are restricted to zero.

■ Example

The application of **lreg** to the above problem is displayed below. Data used in this example consists of 68 observations with 5 input prices. These data have already been normalized around their geometric means. Note that when the Cobb Douglas technology cannot be rejected, the restricted equations (with Cobb Douglas specification and homogeneity in input prices imposed) are constructed to estimate the cost function again.

```
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Program file: translogmt.e
Data set:     translogmt
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/

library lrmt;
#include lrmt.sdf

struct lrControl lrc;
struct lrOut lro;
lrc = lrControlCreate;
output file = translogmt.out reset;

dataloop translogmt newdatamt;    /* Perform data transformation */
   cost = ln(cost);
   y = ln(y);
   p1 = ln(p1);
   p2 = ln(p2);
   p3 = ln(p3);
   p4 = ln(p4);
   p5 = ln(p5);
   make yy = (y*y)/2;
   make p11 = (p1*p1)/2;
   make p12 = p1*p2;
   make p13 = p1*p3;
   make p14 = p1*p4;
   make p15 = p1*p5;
   make p22 = (p2*p2)/2;
   make p23 = p2*p3;
   make p24 = p2*p4;
   make p25 = p2*p5;
   make p33 = (p3*p3)/2;
   make p34 = p3*p4;
   make p35 = p3*p5;
   make p44 = (p4*p4)/2;
   make p45 = p4*p5;
   make p55 = (p5*p5)/2;
   make yp1 = y*p1;
   make yp2 = y*p2;
   make yp3 = y*p3;
   make yp4 = y*p4;
   make yp5 = y*p5;
endata;

dataset = "newdatamt";
dv = "cost";
string iv = { "const","y","p1","p2","p3","p4","p5",
            "yy",
            "p11","p12","p13","p14","p15",
            "p22","p23","p24","p25",
            "p33","p34","p35",
            "p44","p45",
            "p55",
            "yp1","yp2","yp3","yp4","yp5" };
lro = lreg(lrc,dataset,dv,iv,0);

/* Test of constant return to size */
test1 = "y=1, yy=0, yp1=0, yp2=0, yp3=0, yp4=0, yp5=0";
call lrtest(lrc,lro,test1);

/* Test of Cobb Douglas technology */
test2 = "yy=0,
         p11=0, p12=0, p13=0, p14=0, p15=0,
         p22=0, p23=0, p24=0, p25=0,
         p33=0, p34=0, p35=0,
         p44=0, p45=0,
         p55=0,
         yp1=0, yp2=0, yp3=0, yp4=0, yp5=0";
call lrtest(lrc,lro,test2);

lrc.title = "COBB DOUGLAS TECHNOLOGY AND HOMOGENEITY IMPOSED";
restrict = "p1+p2+p3+p4+p5=1" $+ "," $+ test2;  /* note here */
call lreg(lrc,dataset,dv,iv,restrict);

output off;

/*+++++  end of program file  +++++++++++++++++++++++++++++++++*/
```

Output for this example:

```
==============================================================================
 LINEAR REGRESSION Version 1.0.0                            6/08/2004   3:56 pm
==============================================================================
                            Data Set:  newdatamt
------------------------------------------------------------------------------


                  ----------------------------------
                     Dependent variable:      cost
                  ----------------------------------

Total cases:                    68      Valid cases:                        68
Total SS:                   34.870      Degrees of freedom:                 40
R-squared:                   0.790      Rbar-squared:                    0.648
Residual SS:                 7.336      Std error of est:                0.428
F(27,40):                    5.560      Probability of F:                0.000
Durbin-Watson:               1.920

                          Standard                 Prob   Standardized  Cor with
Variable     Estimate      Error       t-value     >|t|     Estimate    Dep Var
------------------------------------------------------------------------------
const       -0.743055    0.637914    -1.164819     0.251   -0.000000    0.000000
y            0.503951    0.078073     6.454869     0.000    0.652219    0.792476
p1           0.133066    0.083615     1.591409     0.119    0.587181    0.331334
p2           0.105448    0.160177     0.658322     0.514    0.083346   -0.064546
p3           0.346294    0.232256     1.491002     0.144    0.145411    0.195297
p4          -0.422912    0.817184    -0.517523     0.608   -0.195221   -0.002283
p5          -0.227474    0.205070    -1.109250     0.274   -0.140038    0.129591
yy           0.270977    0.130397     2.078093     0.044    0.218877    0.385657
p11          0.132195    0.120351     1.098415     0.279    0.397203   -0.182481
p12         -0.081801    0.060569    -1.350541     0.184   -0.181723   -0.066657
p13         -0.097157    0.082348    -1.179842     0.245   -0.134457    0.011821
```

```
p14        -0.107957    0.086415   -1.249290     0.219   -0.170307   -0.164789
p15         0.063802    0.052807    1.208222     0.234    0.119138    0.127592
p22         0.041148    0.321014    0.128183     0.899    0.027709    0.027746
p23         0.134335    0.726283    0.184962     0.854    0.020963   -0.045316
p24        -0.204401    0.495043   -0.412896     0.682   -0.061939    0.027906
p25         0.194994    0.412987    0.472155     0.639    0.062974   -0.057050
p33         0.240103    1.075696    0.223207     0.825    0.024031   -0.009060
p34        -0.013950    0.997745   -0.013981     0.989   -0.001986   -0.018197
p35         0.360854    0.759751    0.474963     0.637    0.068868   -0.049615
p44        -1.563439    2.446582   -0.639030     0.526   -0.246312   -0.015852
p45         1.401099    0.950206    1.474521     0.148    0.272823   -0.169663
p55         0.029768    0.444437    0.066979     0.947    0.009755   -0.019073
yp1         0.005353    0.027006    0.198204     0.844    0.020575    0.216620
yp2         0.090026    0.299039    0.301052     0.765    0.056609   -0.062936
yp3         0.338722    0.282181    1.200371     0.237    0.127290    0.242308
yp4        -0.364306    0.277432   -1.313137     0.197   -0.152585   -0.123340
yp5         0.091478    0.293100    0.312106     0.757    0.036010    0.229466

*** Test of Constant Returns to Size

-----  LREG:  Results for Linear Hypothesis Testing  --------------------------

      F(7,40) statistic =         7.596      Prob. =  0.000

-------------------------------------------------------------------------------

*** Test of Cobb Douglas Technology

-----  LREG:  Results for Linear Hypothesis Testing  --------------------------

      F(21,40) statistic =        0.940      Prob. =  0.548

-------------------------------------------------------------------------------

==============================================================================
                COBB DOUGLAS TECHNOLOGY AND HOMOGENEITY IMPOSED
==============================================================================
 LINEAR REGRESSION Version 1.0.0                          6/08/2004   3:56 pm
==============================================================================
                             Data Set:  newdatamt
------------------------------------------------------------------------------

RESTRICTIONS IN EFFECT

                   ----------------------------------
                      Dependent variable:      cost
                   ----------------------------------

Total cases:                     68      Valid cases:                       68
Total SS:                    34.870      Degrees of freedom:                62
```

```
R-squared:                   0.654     Rbar-squared:                   0.626
Residual SS:                12.067     Std error of est:               0.441
F(5,62):                    23.433     Probability of F:               0.000
Durbin-Watson:               2.128

                         Standard                  Prob   Standardized  Cor with
Variable     Estimate      Error       t-value     >|t|     Estimate    Dep Var
-------------------------------------------------------------------------------
const        0.000000    0.053499    0.000000     1.000    0.000000    0.000000
y            0.588345    0.061114    9.627047     0.000    0.761442    0.792476
p1           0.033788    0.018047    1.872239     0.066    0.149095    0.331334
p2           0.176374    0.087734    2.010317     0.049    0.139406   -0.064546
p3           0.640062    0.162847    3.930453     0.000    0.268765    0.195297
p4           0.064513    0.137881    0.467890     0.642    0.029780   -0.002283
p5           0.085264    0.122994    0.693239     0.491    0.052491    0.129591
yy          -0.000000    0.000000   -0.000000     1.000   -0.000000    0.385657
p11         -0.000000    0.000000      -----      -----   -0.000000   -0.182481
p12         -0.000000    0.000000   -0.000000     1.000   -0.000000   -0.066657
p13          0.000000    0.000000    0.000000     1.000    0.000000    0.011821
p14         -0.000000    0.000000   -0.000000     1.000   -0.000000   -0.164789
p15         -0.000000    0.000000   -0.000000     1.000   -0.000000    0.127592
p22          0.000000    0.000000    0.000000     1.000    0.000000    0.027746
p23         -0.000000    0.000000   -0.000000     1.000   -0.000000   -0.045316
p24         -0.000000    0.000000   -0.000000     1.000   -0.000000    0.027906
p25         -0.000000    0.000000      -----      -----   -0.000000   -0.057050
p33          0.000000    0.000000    0.000000     1.000    0.000000   -0.009060
p34          0.000000    0.000000    0.000000     1.000    0.000000   -0.018197
p35         -0.000000    0.000000   -0.000000     1.000   -0.000000   -0.049615
p44         -0.000000    0.000000   -0.000000     1.000   -0.000000   -0.015852
p45          0.000000    0.000000    0.000000     1.000    0.000000   -0.169663
p55          0.000000    0.000000    0.000000     1.000    0.000000   -0.019073
yp1         -0.000000    0.000000   -0.000000     1.000   -0.000000    0.216620
yp2          0.000000    0.000000    0.000000     1.000    0.000000   -0.062936
yp3          0.000000    0.000000    0.000000     1.000    0.000000    0.242308
yp4          0.000000    0.000000    0.000000     1.000    0.000000   -0.123340
yp5          0.000000    0.000000    0.000000     1.000    0.000000    0.229466
```

By looking at the results, the test of constant returns to size is rejected. However, the Cobb Douglas technology cannot be rejected at even 1% level of significance. Therefore, the cost function is estimated again with the Cobb Douglas technology and homogeneity in input prices imposed. Alternatively, you can estimate the Cobb Douglas functional form as below.

$$\ln C = \alpha_0 + \alpha_y \ln Y + \sum_i \alpha_i \ln P_i$$

With homogeneity imposed ($\sum_i \alpha_i = 1$), both estimations should give identical results. You can confirm this by trying the following lines in the command file.

```
restrict = "p1+p2+p3+p4+p5=1";
dv = "cost";
iv = "const"$|"y"$|"p1"$|"p2"$|"p3"$|"p4"$|"p5";
call lreg(lrc,dataset,dv,iv,restrict);
```

3.4 Estimating a System of Cost Share Equations Using Seemingly Unrelated Regression

This section demonstrates the use of **lsur** to estimate the system of cost shares with seemingly unrelated regression technique. Linear hypothesis testing and restrictions imposed on the parameters are demonstrated as well.

The system of cost shares are defined as follows:

$$S_{it} = \alpha_i + \sum_j \alpha_{ij} \ln P_{jt} + \gamma_i \ln Y_t + \beta_i trend + \varepsilon_{it} \qquad i = 1, 2, 3, 4$$

where S_i are the cost shares and derived from the translog cost function, *trend* is a time trend (that is, $t = 1$ for the first observation, $t = 2$ for the second observation, and so on), P_j is the j^{th} input price, and Y is the output.

There are several hypotheses of interest:

1. Symmetry $\alpha_{ij} = \alpha_{ji} \ \forall \ i \neq j$
2. Homogeneity $\sum_j \alpha_{ij} = 0 \ \forall \ i$ and $\sum_i \alpha_i = 1$
3. Constant returns to scale $\gamma_i \ \forall \ i$
4. No technical change $\beta_i \ \forall \ i$

Besides the above hypotheses, you may want to estimate the systems with (1) and (2) above imposed. Since the shares must add to unity, one equation must be dropped to prevent a singular variance-covariance matrix . Kmenta and Gilbert [11] have shown that the *Iterative Seemingly-Unrelated Regression* can produce asymptotically maximum likelihood estimates. The parameter estimates are the same whichever equation is deleted.

Example

The program file for this problem is `sharesmt.e`. This system model has four input prices, thus it has four equations. However, because of the problem of the singular variance-covariance matrix, the 4^{th} equation is dropped from the model. To illustrate that it is irrelevant which equation is dropped, the final model reestimates with the 4^{th} equation included and the 1^{st} equation excluded. Data for this example are put into two files, `sharemt.dat` and `pricemt.dat`. Although the merging of two data files is not presently available in **dataloop**, you can use the **GAUSS** language to implement this. If you have difficulty seeing how the restrictions are constructed, try to write out the share equations. More details of the **lsur** procedure can be found in the command reference of the next chapter.

```
/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Program file: sharesmt.e
Data set:     sharemt
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/

/*  THIS PART IS TO COMBINE TWO DATA SETS  */
open f1 = share;
open f2 = price;
vnames = getnamef(f1)$|getnamef(f2);
create fout = newdatamt with ^vnames,0,8;

do until eof(f1);
   data = readr(f1,100)~readr(f2,100);
   data = data[.,1:4]~ln(data[.,5:10])~data[.,11];
   if writer(fout,data) /= rows(data);
      print "disk full"; end;
   endif;
endo;
closeall;

library lrmt;
#include lrmt.sdf

struct lrControl lrc;
struct lrOut lro;
lrc = lrControlCreate;
output file = sharesmt.out reset;

y = "s1"$|"s2"$|"s3";
string x = { "const","p1","p2","p3","p4","y","trend",   /* 1st Eqn. */
             "const","p1","p2","p3","p4","y","trend",   /* 2nd Eqn. */
             "const","p1","p2","p3","p4","y","trend" }; /* 3rd Eqn. */
novars = { 7,7,7 };     /* No. of RHS variables in each equation */

lro = lsur(lrc,"newdatamt",y,x,novars,0);

print "TEST OF SYMMETRY ";
```

```
test1 = "p2:1-p1:2=0, p3:1-p1:3=0, p3:2-p2:3=0";
call lrtest(lrc,lro,test1);

print "TEST OF HOMOGENEITY ";
test2 = "p1:1 + p2:1 + p3:1 + p4:1 = 0,
         p1:2 + p2:2 + p3:2 + p4:2 = 0,
         p1:3 + p2:3 + p3:3 + p4:3 = 0";
call lrtest(lrc,lro,test2);

print "TEST OF CONSTANT RETURNS TO SCALE ";
test3 = "y:1=0, y:2=0, y:3=0";
call lrtest(lrc,lro,test3);

print "TEST OF NO TECHNICAL CHANGE ";
test4 = "trend:1=0, trend:2=0, trend:3=0";
call lrtest(lrc,lro,test4);

lrc.tol = 0.00000001;
lrc.iter = 100;
lrc.title = "SYMMETRY AND HOMOGENEITY IMPOSED USING S1,S2,S3";
restrict = "p2:1-p1:2=0,
            p3:1-p1:3=0,
            p3:2-p2:3=0,
            p1:1 + p2:1 + p3:1 + p4:1 = 0,
            p1:2 + p2:2 + p3:2 + p4:2 = 0,
            p1:3 + p2:3 + p3:3 + p4:3 = 0";
call lsur(lrc,"newdatamt",y,x,novars,restrict);

/*  USING S4 AND REMOVING S1 */
y = "s2"$|"s3"$|"s4";
string x = { "const","p1","p2","p3","p4","y","trend",
      "const","p1","p2","p3","p4","y","trend",
      "const","p1","p2","p3","p4","y","trend" };
novars = { 7,7,7 };
lrc.tol = 0.00000001;
lrc.iter = 100;
lrc.title = "SYMMETRY AND HOMOGENEITY IMPOSED USING S2,S3,S4";
restrict = "p3:1-p2:2=0,
            p4:1-p2:3=0,
            p4:2-p3:3=0,
            p1:1 + p2:1 + p3:1 + p4:1 = 0,
            p1:2 + p2:2 + p3:2 + p4:2 = 0,
            p1:3 + p2:3 + p3:3 + p4:3 = 0";
call lsur(lrc,"newdatamt",y,x,novars,restrict);

output off;
```

```
/*+++++  end of program file  ++++++++++++++++++++++++++++++++++*/
```

Output for the example:

From the output below, there are two interesting results. First, all of the hypotheses are rejected at any level of significance. It seems to be somewhat disappointing owing to the violation of the economic theory (i.e., the conditions of symmetry and homogeneity). However, these conditions are seldom tested. In fact, according to Young et al. [17] in most previous studies that have used the flexible functional forms, the properties of the cost function such as the curvature condition is either not tested or rejected. This applies as well to the homogeneity condition. In standard practice, the symmetry and homogeneity are imposed in an ad hoc manner. Second, you can confirm that the *Iterative Seemingly Unrelated Regression* can produce asymptotically maximum likelihood estimates (i.e., obtained parameter estimates are invariant which respect to which equation is deleted) by excluding a different equation.

```
================================================================================
 LINEAR SEEMINGLY UNRELATED REGRESSION Version 1.0.0       6/08/2004   1:12 pm
================================================================================
                             Data Set:  newdatamt
--------------------------------------------------------------------------------


DIVISOR USING N IN EFFECT

        ITER. # =    0    LOG OF DETERMINANT OF SIGMA =   -28.63610569
        ITER. # =    1    LOG OF DETERMINANT OF SIGMA =   -28.63610569

                   -----------------------------------
                            Equation:    1
                    Dependent variable:         s1
                   -----------------------------------

     Total cases:             51    Valid cases:                    51
     Total SS:             0.143    Degrees of freedom:           ----
     R-squared:            0.973    Rbar-squared:                0.970
     Residual SS:          0.004    Std error of est:            0.009
     Durbin-Watson:        0.866

                    Estimated        Standard                    Prob
     Variable      Coefficient         Error        t-ratio      >|t|
     -------------------------------------------------------------------
     const           0.42061914      0.03214251      13.086      0.0000
     p1              0.40079591      0.01905988      21.028      0.0000
     p2             -0.06503234      0.03126903      -2.080      0.0434
     p3             -0.00663223      0.02213495      -0.300      0.7659
     p4             -0.01302998      0.01050634      -1.240      0.2215
     y             -0.35219294       0.04038780       -8.720       0.0000
     trend          0.00064332       0.00086750        0.742       0.4623


                   ----------------------------------
                              Equation:    2
                          Dependent variable:        s2
                   ----------------------------------

     Total cases:                51    Valid cases:                    51
     Total SS:                0.110    Degrees of freedom:           ----
     R-squared:               0.875    Rbar-squared:                0.858
     Residual SS:             0.014    Std error of est:            0.016
     Durbin-Watson:           0.496

                    Estimated        Standard                      Prob
     Variable      Coefficient         Error        t-ratio       >|t|
     ------------------------------------------------------------------
     const          0.25782323       0.06079927        4.241       0.0001
     p1            -0.42352146       0.03605278      -11.747       0.0000
     p2            -0.00105493       0.05914705       -0.018       0.9859
     p3            -0.05348148       0.04186944       -1.277       0.2082
     p4            -0.04980891       0.01987330       -2.506       0.0160
     y              0.50219016       0.07639569        6.574       0.0000
     trend          0.00112845       0.00164093        0.688       0.4953


                   ----------------------------------
                              Equation:    3
                          Dependent variable:        s3
                   ----------------------------------

     Total cases:                51    Valid cases:                    51
     Total SS:                0.044    Degrees of freedom:           ----
     R-squared:               0.939    Rbar-squared:                0.931
     Residual SS:             0.003    Std error of est:            0.007
     Durbin-Watson:           0.724

                    Estimated        Standard                      Prob
     Variable      Coefficient         Error        t-ratio       >|t|
     ------------------------------------------------------------------
     const         -0.03571263       0.02678922       -1.333       0.1894
     p1            -0.05133372       0.01588548       -3.231       0.0023
     p2            -0.09499125       0.02606122       -3.645       0.0007
     p3             0.01786662       0.01844840        0.968       0.3381
     p4            -0.02318483       0.00875652       -2.648       0.0112
     y              0.10671202       0.03366127        3.170       0.0028
     trend          0.00331548       0.00072302        4.586       0.0000

TEST OF SYMMETRY
```

```
-----  LSUR:  Results for Linear Hypothesis Testing  ---------------------------

       Wald Chi-SQ(3) statistic =  76.032      Prob. =  0.000

-------------------------------------------------------------------------------

TEST OF HOMOGENEITY

-----  LSUR:  Results for Linear Hypothesis Testing  ---------------------------

       Wald Chi-SQ(3) statistic = 108.365      Prob. =  0.000

-------------------------------------------------------------------------------

TEST OF CONSTANT RETURNS TO SCALE

-----  LSUR:  Results for Linear Hypothesis Testing  ---------------------------

       Wald Chi-SQ(3) statistic = 158.533      Prob. =  0.000

-------------------------------------------------------------------------------

TEST OF NO TECHNICAL CHANGE

-----  LSUR:  Results for Linear Hypothesis Testing  ---------------------------

       Wald Chi-SQ(3) statistic =  56.880      Prob. =  0.000

-------------------------------------------------------------------------------

==============================================================================
                SYMMETRY AND HOMOGENEITY IMPOSED USING S1,S2,S3
==============================================================================
 LINEAR SEEMINGLY UNRELATED REGRESSION Version 1.0.0       6/08/2004   1:12 pm
==============================================================================
                             Data Set:  newdatamt
------------------------------------------------------------------------------


DIVISOR USING N IN EFFECT
RESTRICTIONS IN EFFECT

        ITER. # =    0    LOG OF DETERMINANT OF SIGMA =   -27.33021909
        ITER. # =    1    LOG OF DETERMINANT OF SIGMA =   -27.39912399
        ITER. # =    2    LOG OF DETERMINANT OF SIGMA =   -27.39948493
        ITER. # =    3    LOG OF DETERMINANT OF SIGMA =   -27.39948665
        ITER. # =    4    LOG OF DETERMINANT OF SIGMA =   -27.39948666
        ITER. # =    5    LOG OF DETERMINANT OF SIGMA =   -27.39948666
```

```
                  ----------------------------------
                             Equation:    1
                     Dependent variable:        s1
                  ----------------------------------

     Total cases:              51    Valid cases:                  51
     Total SS:              0.143    Degrees of freedom:         ----
     R-squared:             0.954    Rbar-squared:              0.954
     Residual SS:           0.007    Std error of est:          0.011
     Durbin-Watson:         0.729

                    Estimated        Standard                    Prob
     Variable      Coefficient         Error        t-ratio      >|t|
     --------------------------------------------------------------------
     const           0.32351807       0.02125978       15.217       0.0000
     p1              0.29551004       0.01140614       25.908       0.0000
     p2             -0.23185846       0.01478196      -15.685       0.0000
     p3             -0.00831974       0.00834634       -0.997       0.3237
     p4             -0.05533185       0.00707085       -7.825       0.0000
     y              -0.12711952       0.01089636      -11.666       0.0000
     trend           0.00299290       0.00064649        4.629       0.0000


                  ----------------------------------
                             Equation:    2
                     Dependent variable:        s2
                  ----------------------------------

     Total cases:              51    Valid cases:                  51
     Total SS:              0.110    Degrees of freedom:         ----
     R-squared:             0.802    Rbar-squared:              0.802
     Residual SS:           0.022    Std error of est:          0.021
     Durbin-Watson:         0.259

                    Estimated        Standard                    Prob
     Variable      Coefficient         Error        t-ratio      >|t|
     --------------------------------------------------------------------
     const           0.37307088       0.03770198        9.895       0.0000
     p1             -0.23185846       0.01478196      -15.685       0.0000
     p2              0.23773082       0.02815147        8.445       0.0000
     p3             -0.02367121       0.01692764       -1.398       0.1682
     p4              0.01779884       0.01343222        1.325       0.1912
     y               0.10516417       0.01698785        6.191       0.0000
     trend          -0.00149076       0.00113327       -1.315       0.1944


                  ----------------------------------
                             Equation:    3
                     Dependent variable:        s3
                  ----------------------------------
```

```
    Total cases:              51   Valid cases:                   51
    Total SS:              0.044   Degrees of freedom:          ----
    R-squared:             0.926   Rbar-squared:               0.926
    Residual SS:           0.003   Std error of est:           0.008
    Durbin-Watson:         0.648

                 Estimated        Standard                      Prob
    Variable     Coefficient        Error        t-ratio       >|t|
    -------------------------------------------------------------------
    const          0.00535612      0.02658260       0.201       0.8411
    p1            -0.00831974      0.00834634      -0.997       0.3237
    p2            -0.02367121      0.01692764      -1.398       0.1682
    p3             0.03282478      0.01975992       1.661       0.1029
    p4            -0.00083384      0.00634676      -0.131       0.8960
    y              0.00065254      0.00891954       0.073       0.9420
    trend          0.00229339      0.00073096       3.138       0.0029

==============================================================================
                SYMMETRY AND HOMOGENEITY IMPOSED USING S2,S3,S4
==============================================================================
 LINEAR SEEMINGLY UNRELATED REGRESSION Version 1.0.0       6/08/2004   1:12 pm
==============================================================================
                            Data Set:  newdatamt
------------------------------------------------------------------------------


DIVISOR USING N IN EFFECT
RESTRICTIONS IN EFFECT

        ITER. # =    0    LOG OF DETERMINANT OF SIGMA =   -27.26517712
        ITER. # =    1    LOG OF DETERMINANT OF SIGMA =   -27.39866015
        ITER. # =    2    LOG OF DETERMINANT OF SIGMA =   -27.39948790
        ITER. # =    3    LOG OF DETERMINANT OF SIGMA =   -27.39949111
        ITER. # =    4    LOG OF DETERMINANT OF SIGMA =   -27.39949112
        ITER. # =    5    LOG OF DETERMINANT OF SIGMA =   -27.39949112

                   ----------------------------------
                             Equation:    1
                     Dependent variable:        s2
                   ----------------------------------

    Total cases:              51   Valid cases:                   51
    Total SS:              0.110   Degrees of freedom:          ----
    R-squared:             0.802   Rbar-squared:               0.802
    Residual SS:           0.022   Std error of est:           0.021
    Durbin-Watson:         0.259

                 Estimated        Standard                      Prob
    Variable     Coefficient        Error        t-ratio       >|t|
    --------------------------------------------------------------------
    const          0.37306890       0.03770206          9.895        0.0000
    p1            -0.23185685       0.01478201        -15.685        0.0000
    p2             0.23773025       0.02815156          8.445        0.0000
    p3            -0.02367155       0.01692765         -1.398        0.1682
    p4             0.01779815       0.01343223          1.325        0.1912
    y              0.10516289       0.01698791          6.190        0.0000
    trend         -0.00149070       0.00113328         -1.315        0.1944


                   -----------------------------------
                             Equation:    2
                     Dependent variable:         s3
                   -----------------------------------

    Total cases:                51    Valid cases:                     51
    Total SS:                0.044    Degrees of freedom:            ----
    R-squared:               0.926    Rbar-squared:                 0.926
    Residual SS:             0.003    Std error of est:             0.008
    Durbin-Watson:           0.648

                 Estimated         Standard                        Prob
    Variable      Coefficient          Error         t-ratio        >|t|
    --------------------------------------------------------------------
    const          0.00535567       0.02658261          0.201        0.8411
    p1            -0.00831953       0.00834633         -0.997        0.3237
    p2            -0.02367155       0.01692765         -1.398        0.1682
    p3             0.03282497       0.01975992          1.661        0.1029
    p4            -0.00083389       0.00634676         -0.131        0.8960
    y              0.00065231       0.00891954          0.073        0.9420
    trend          0.00229341       0.00073096          3.138        0.0029


                   -----------------------------------
                             Equation:    3
                     Dependent variable:         s4
                   -----------------------------------

    Total cases:                51    Valid cases:                     51
    Total SS:                0.110    Degrees of freedom:            ----
    R-squared:               0.918    Rbar-squared:                 0.918
    Residual SS:             0.009    Std error of est:             0.013
    Durbin-Watson:           0.328

                 Estimated         Standard                        Prob
    Variable      Coefficient          Error         t-ratio        >|t|
    --------------------------------------------------------------------
    const          0.29805475       0.02261700         13.178        0.0000
    p1            -0.05533192       0.00707086         -7.825        0.0000
    p2             0.01779815       0.01343223          1.325        0.1912
```

```
p3             -0.00083389        0.00634676         -0.131        0.8960
p4              0.03836766        0.00942412          4.071        0.0002
y               0.02130335        0.00941229          2.263        0.0280
trend          -0.00379559        0.00068982         -5.502        0.0000
```

3.5 Estimating Klein's Model I Using Three-Stage Least Squares

This example uses the Klein's Model I [10] for illustration of two- and three-stage least squares. The three behavioral equations are:

$$C = \alpha_0 + \alpha_1 P + \alpha_2 P_{-1} + \alpha_3 (W_p + W_g) + \varepsilon_1 \tag{1}$$
$$I = \beta_0 + \beta_1 P + \beta_2 P_{-1} + \beta_3 K_{-1} + \varepsilon_2 \tag{2}$$
$$W_p = \gamma_0 + \gamma_1 X + \gamma_2 X_{-1} + \gamma_3 A + \varepsilon_3 \tag{3}$$

Equations 1 to 3 are respectively the consumption equation, investment equation, and demand-for-labour equation; where

C	=	Consumption
P	=	Profits
P_{-1}	=	Profits lagged one year
W_p	=	Private wage bill
W_g	=	Government wage bill
K_{-1}	=	Capital stock at the beginning of the year
Y	=	National income
T	=	Indirect taxes
X	=	$Y + T - W_g$
A	=	Time trend measured as years from 1931

The model is completed by the following three identities:

$$\begin{aligned} Y + T &= C + I + G \\ Y &= W_p + W_g + P \\ K &= K_{-1} + I \end{aligned}$$

where G is government spending on goods and services.

This above system includes six endogenous variables (C,P,W_p, I,Y,K) and seven predetermined variables (W_g,T,G,A, P_{-1},K_{-1},$(Y+T-W_g)_{-1}$). All three behavioral equations are overidentified. According to Zellner and Theil [18], the three identities should be removed from the estimation.

Example

From the data set, some variables such as the lagged variables and the time trend are not available. Hence, we demonstrate the use of the **dataloop** to create a new data set. Inside the **dataloop** the **lag**'s are used to create the lagged variables. Iteration of the estimation process and parameter restriction across equations are available inside the **l3sls** procedure. Details of these can be found in the command reference of the next chapter.

```
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Program file: kleinmt.e
Data set:     kleinmt
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/

library lrmt;
#include lrmt.sdf

struct lrControl lrc;
struct lrOut lro;
lrc = lrControlCreate;

dataloop kleinmt newdatamt;    /* Generate new data set */
   make wsum = wp + wg;
   make trend = year - 1931;
   lag klag = k:1;
   lag plag = p:1;
   lag xlag = x:1;
   keep year c p wp i x wg g t k wsum trend klag plag xlag;
endata;

output file = kleinmt.out reset;
lhs = "c"$|"i"$|"wp";           /* LHS variables for the model */
string rhs = { "const","p","plag","wsum",  /* RHS variables for 1st equation */
        "const","p","plag","klag",         /* 2nd equation */
        "const","x","xlag","trend" };      /* 3rd equation */

/* Exogenous variables */
string exo = { "const","wg","t","g","trend","plag","klag","xlag" };
novars = { 4,4,4 };    /* No. of RHS variables in each eqn. */
```

```
lrc.dv = 0;               /* Use the normal divisor */
lro = l3sls(lrc,"newdatamt",lhs,rhs,exo,novars,0);

output off;

/*++++++  end of program file  +++++++++++++++++++++++++++++++++++*/
```

Here is the output:

```
==============================================================================
 LINEAR THREE-STAGE LEAST SQUARES Version 1.0.0          6/08/2004   2:15 pm
==============================================================================
                            Data Set:  newdatamt
------------------------------------------------------------------------------


*************************  TWO-STAGE RESULTS  ********************************


                  ----------------------------------
                            Equation:    1
                    Dependent variable:         C
                  ----------------------------------

     Total cases:              22    Valid cases:                  21
     Total SS:            941.430    Degrees of freedom:           17
     R-squared:             0.977    Rbar-squared:              0.973
     Residual SS:          21.925    Std error of est:          1.136
     Durbin-Watson:         1.485

                   Estimated        Standard                      Prob
     Variable     Coefficient         Error        t-ratio       >|t|
     ------------------------------------------------------------------
     CONST        16.55475577       1.46797870      11.277       0.0000
     P             0.01730221       0.13120458       0.132       0.8966
     PLAG          0.21623404       0.11922168       1.814       0.0874
     WSUM          0.81018270       0.04473506      18.111       0.0000


                  ----------------------------------
                            Equation:    2
                    Dependent variable:         I
                  ----------------------------------

     Total cases:              22    Valid cases:                  21
     Total SS:            252.327    Degrees of freedom:           17
     R-squared:             0.885    Rbar-squared:              0.865
```

```
Residual SS:           29.047    Std error of est:            1.307
Durbin-Watson:          2.085

               Estimated         Standard                      Prob
Variable      Coefficient          Error         t-ratio       >|t|
-------------------------------------------------------------------
CONST         20.27820894       8.38324890         2.419       0.0271
P              0.15022182       0.19253359         0.780       0.4460
PLAG           0.61594358       0.18092585         3.404       0.0034
KLAG          -0.15778764       0.04015207        -3.930       0.0011


              ----------------------------------
                        Equation:    3
                 Dependent variable:         WP
              ----------------------------------

Total cases:               22    Valid cases:                   21
Total SS:             794.910    Degrees of freedom:            17
R-squared:              0.987    Rbar-squared:               0.985
Residual SS:           10.005    Std error of est:           0.767
Durbin-Watson:          1.963

               Estimated         Standard                      Prob
Variable      Coefficient          Error         t-ratio       >|t|
-------------------------------------------------------------------
CONST          1.50029689       1.27568637         1.176       0.2558
X              0.43885907       0.03960266        11.082       0.0000
XLAG           0.14667382       0.04316395         3.398       0.0034
TREND          0.13039569       0.03238839         4.026       0.0009


*************************  THREE-STAGE RESULTS  ********************************


   ITER. # =    0    LOG OF DETERMINANT OF SIGMA =    -0.61186241
   ITER. # =    1    LOG OF DETERMINANT OF SIGMA =    -0.62839294

              ----------------------------------
                        Equation:    1
                 Dependent variable:          C
              ----------------------------------

Total cases:               22    Valid cases:                   21
Total SS:             941.430    Degrees of freedom:            17
R-squared:              0.980    Rbar-squared:               0.977
Residual SS:           18.727    Std error of est:           1.050
Durbin-Watson:          1.425

               Estimated         Standard                      Prob
```

```
Variable      Coefficient         Error        t-ratio       >|t|
-------------------------------------------------------------------
CONST         16.44079006      1.44992488       11.339       0.0000
P              0.12489047      0.12017872        1.039       0.3133
PLAG           0.16314409      0.11163081        1.461       0.1621
WSUM           0.79008094      0.04216562       18.738       0.0000


              ----------------------------------
                         Equation:    2
                 Dependent variable:         I
              ----------------------------------

Total cases:              22    Valid cases:                     21
Total SS:            252.327    Degrees of freedom:              17
R-squared:             0.826    Rbar-squared:                 0.795
Residual SS:          43.954    Std error of est:             1.608
Durbin-Watson:         1.996

               Estimated        Standard                       Prob
Variable      Coefficient         Error        t-ratio       >|t|
-------------------------------------------------------------------
CONST         28.17784687      7.55085338        3.732       0.0017
P             -0.01307918      0.17993761       -0.073       0.9429
PLAG           0.75572396      0.16997567        4.446       0.0004
KLAG          -0.19484825      0.03615585       -5.389       0.0000


              ----------------------------------
                         Equation:    3
                 Dependent variable:        WP
              ----------------------------------

Total cases:              22    Valid cases:                     21
Total SS:            794.910    Degrees of freedom:              17
R-squared:             0.986    Rbar-squared:                 0.984
Residual SS:          10.921    Std error of est:             0.801
Durbin-Watson:         2.155

               Estimated        Standard                       Prob
Variable      Coefficient         Error        t-ratio       >|t|
-------------------------------------------------------------------
CONST          1.79721773      1.24020347        1.449       0.1655
X              0.40049188      0.03535863       11.327       0.0000
XLAG           0.18129101      0.03796536        4.775       0.0002
TREND          0.14967412      0.03104828        4.821       0.0002
```

Chapter 4

Linear Regression MT Reference

A summary table listing all of the procedures is displayed below.

Procedure	*Description*	*Page*
l2sls	Linear Two-stage Least Squares Estimation	40
l3sls	Linear Three-stage Least Squares Estimation	45
lrControlCreate	Sets Default Values	51
lreg	Ordinary Least Squares Estimation	52
lrerror	Error Handling Procedure	61
lrtest	Performs Linear Hypothesis Testing	63
lsur	Linear Seemingly Unrelated Regression	67
Rmatrix	Constructs Restriction Matrix	74
SRmatrix	Constructs System Restriction Matrix	76

Purpose

l2sls (Linear Two-Stage Least Squares) is a single equation technique which employs the generalized least squares rules for estimating equations in a simultaneous equations model.

Library

lrmt

Format

lro = **l2sls(***lrc*,*dataset*,*LHS_var*,*RHS_vars*,*EXO_vars*,*Restrict***);**

Input

lrc an instance of an **lrControl** structure. The following members of *lrc* are referenced within the **l2sls** routine:

lrc.`dv` scalar, determines which divisor is used to compute $\hat{\sigma}_{jj}$.

- **0** $(T - K)$ is used as divisor, where T is the number of observations, K is the number of all right-hand side variables in the equation.
- **1** T is used as divisor - this provides good asymptotic properties for the estimator when the sample size is large.

Default = 1.

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

t	print title (see *lrc*.`title`)
l	bracket title with lines
d	print date and time
v	print procedure name and version number
f	print file name being analyzed

Example:

```
lrc.header = "tld";
```

If *lrc*.`header` == “”, no header is printed. Default = “tldvf”.

lrc.`output` determines printing of intermediate results.

- **0** nothing is written.
- **1** serial ASCII output format suitable for disk files or printers.

Default = 1.

lrc.`pcor` scalar, if 1, print the correlation matrix of coefficients. This is the $\left[Z_j'X(X'X)^{-1}X'Z_j\right]^{-1}$ matrix scaled to unit diagonals and is *not* the correlation matrix of variables. Default = 0.

lrc.`pcov` scalar, if 1, print the covariance matrix of coefficients which is $\hat{\sigma}_{jj}\left[Z_j^{'}X(X'X)^{-1}X'Z_j\right]^{-1}$, where $\hat{\sigma}_{jj}$ is the mean squared error. Default = 0.

lrc.`range` 2×1 vector, specifies the range of the data set to be used in estimation. The first element specifies the beginning observation while the second element specifies the ending observation. For example: *lrc.*`range`={ 100,200 }. Default is { 0,0 } and uses the whole data set.

lrc.`ranktol` scalar, specifies the tolerance used to determine if any of the singular values are effectively 0 when computing the rank of a matrix. Default = 10^{-13}.

lrc.`row` scalar, specifies how many rows of the data set are read per iteration of the read loop. If *lrc.*`row` = 0, the number of rows to be read is calculated by the program. Default = 0.

lrc.`rowfac` scalar, “row factor”. If an **LRMT** procedure fails due to insufficient memory while attempting to read a **GAUSS** data set, then *lrc.*`rowfac` may be set to some value between 0 and 1 to read a *proportion* of the original number of rows of the **GAUSS** data set. For example, setting

```
lrc.rowfac = 0.8;
```

causes **GAUSS** to read in only 80% of the rows that were originally calculated. This global has an effect only when *lrc.*`row` = 0. Default = 1.

lrc.`title` string, message printed at the top of the results. Default = “”.

dataset string, name of **GAUSS** data set.

LHS_var string, name of the endogenous variable in the equation.

RHS_vars $K \times 1$ string array, all the right-hand side variables in the equation. If a constant vector is desired, simply put “CONST” in the *RHS_vars* list.

EXO_vars $P \times 1$ string array, all the exogenous variables in the system. Specify “CONST” in the *EXO_vars* list should a constant vector be desired.

Restrict string or 0, if *Restrict* equals 0, estimation without restrictions is performed. Otherwise, the estimator is estimated with the given restrictions. The syntax of *Restrict* is as follows:

Restrict = “$restriction1, restriction2, \cdots, restrictionJ$”;

More than one restriction is allowed provided each is separated by a comma. Each restriction must be written as a linear equation with all variables on the left hand side and the constant on the right hand side (i.e., $x1 + x2 = 1$). Variables shown in each restriction must be variables in the right-hand side of the equation. Restrictions in the *Restrict* argument must be consistent and not redundant otherwise error messages occur. *You should make sure that only the parameters associated with the variables are restricted and not the variables in the model themselves.*

Example:

```
restrict = "plag - p = 0";
```

▪ Output

lro an instance of an **lrOut** structure, which contains all calculated statistics. The following members of *lro* are set by the **l2sls** routine.

lro.`model`	string, name of the estimation procedure.
lro.`nms`	$K \times 1$ string array, names of the regressors.
lro.`b`	$K \times 1$ vector, regression coefficients.
lro.`vc`	$K \times K$ matrix, variance-covariance matrix of the coefficients.
lro.`se`	$K \times 1$ vector, standard errors of the coefficients.
lro.`s2`	scalar, variance of the estimate ($\hat{\sigma}^2$).
lro.`cx`	$K \times K$ matrix, correlation matrix of the coefficients.
lro.`rsq`	scalar, R^2.
lro.`rbsq`	scalar, adjusted R^2.
lro.`dw`	scalar, Durbin-Watson statistic.
lro.`sse`	scalar, residual sum of squares.
lro.`nobs`	scalar, number of observations.
lro.`ixtx`	$P \times P$ matrix, $(X'X)^{-1}$ as defined in equation (1).
lro.`xtz`	$P \times K$ matrix, $X'Z_j$ as defined in equation (1).
lro.`xty`	$P \times 1$ vector, $X'Y_j$ as defined in equation (1).
lro.`errcode`	scalar, zero or scalar error code.

If errors are encountered, they are handled with the low order bit of the trap flag.

trap 0	terminate with error message
trap 1	return scalar error code in *lro*.`errcode`

For more details on **trap**, see the *COMMAND REFERENCE* of the **GAUSS** manual. Since the returning error code appears as a missing value, it can be translated with the command **scalerr**(*lro*.`errcode`) or viewed with the **lrerror** procedure. See the **lrerror** procedure for more details. Definitions of the error codes can be found in Section 2.6.2 of this manual.

Remarks

l2sls is applicable to equations which are overidentified or exactly identified. Note that **l2sls** provides identical estimates as those of the Indirect Least Squares (ILS) when equations are just identified. However, for an overidentified equation, the ILS can not be used. Instead, the usual alternative is the Two-Stage Least Squares. Good references can be found in Judge, Hill, Griffiths, Lütkepohl, and Lee [8], Judge, Griffiths, Hill, Lütkepohl, and Lee [9], Greene [6], and Johnston [7].

The **l2sls** estimator and its asymptotic variance are as follows:

$$\hat{\delta}_{j,l2sls} = \left[Z_j^{'}X(X'X)^{-1}X'Z_j\right]^{-1}\left[Z_j^{'}X(X'X)^{-1}X'Y_j\right] \tag{1}$$

$$AVAR(\hat{\delta}_{j,l2sls}) = \hat{\sigma}_{jj}\left[Z_j^{'}X(X'X)^{-1}X'Z_j\right]^{-1} \tag{2}$$

where

$$\hat{\sigma}_{jj} = \frac{(Y_j - Z_j\hat{\delta}_{j,l2sls})'(Y_j - Z_j\hat{\delta}_{j,l2sls})}{T} \tag{3}$$

and X is the matrix of all exogenous variables in the system, Y_j and Z_j are the endogenous variable and the right-hand side variables respectively in the j^{th} equation, and T is the total number of observations.

Note:

1. You must be able to specify which variables are endogenous and which are exogenous.
2. $\hat{\delta}_{j,l2sls}$ can be viewed as an instrumental variables estimator with the set of instruments $(X(X'X)^{-1}X'Z_j)$.
3. The denominator in calculating the $\hat{\sigma}_{jj}$ is T and it provides good asymptotic properties for the estimator. In case of small samples, you may choose $T - K_j$ as the divisor, where K_j is the number of right-hand side variables in the j^{th} equation, instead of T. This is accomplished by changing the **lrControl** structure member `dv` to 0.
4. Linear hypothesis testing can be tested with the **lrtest** procedure. However, the distribution is unknown and the test statistic can only be viewed as being asymptotically Chi-Square.
5. Linear a priori restrictions can be imposed on the estimated coefficients.

6. R^2 calculated here is not well defined and could be negative.

Missing data are handled automatically. That is, any observation which has a missing value for any variable is removed from computation.

Example

The following example is from Judge, Hill, Griffiths, Lütkepohl, and Lee [8, page 656].

```
/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Program file: lrmt11.e
Data set:     tmt15_1
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/

library lrmt;
#include lrmt.sdf

struct lrControl lrc;
struct lrOut lro1, lro2, lro3;
lrc = lrControlCreate;

output file = lrmt11.out reset;

dataset = "tmt15_1";

lrc.dv = 0;
x = "x1"$|"x2"$|"x3"$|"x4"$|"x5";

y = "y1";                     /* First equation */
z = "x1"$|"y2"$|"y3";
lro1 = l2sls(lrc,dataset,y,z,x,0);

y = "y2";                     /* Second equation */
z = "x1"$|"y1"$|"x2"$|"x3"$|"x4";
lro2 = l2sls(lrc,dataset,y,z,x,0);

y = "y3";                     /* Third equation */
z = "x1"$|"y2"$|"x2"$|"x5";
lro3 = l2sls(lrc,dataset,y,z,x,0);

output off;

/*++++++  end of program file  +++++++++++++++++++++++++++++++++++*/
```

Source

`l2slsmt.src`

Purpose

l3sls (Linear Three-Stage Least Squares) is a procedure for the estimation of the parameters of a system of simultaneous equations. A synopsis of the estimation procedure is found in Judge, Hill, Griffiths, Lütkepohl, and Lee [8, Ch. 15], Greene [6, Ch. 19], and Johnston [7, Ch. 11].

Library

lrmt

Format

lro = **l3sls(***lrc*,*dataset*,*LHS_vars*,*RHS_vars*,*EXO_vars*,*Novars*,*Restrict***);**

Input

lrc an instance of an **lrControl** structure. The following members of *lrc* are referenced within the **l3sls** routine:

lrc.`dv` scalar, determines which divisor is used to compute $\hat{\sigma}_{ij}$.

0 $(MT - K)/M$ is used as divisor, where M is the number of equations, T is the number of observations, and K is the number of all estimated parameters in the model.

1 T is used as divisor. This will provide good asymptotic properties for the estimator when the sample size is large.

Default = 1.

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

t	print title (see *lrc*.`title`)
l	bracket title with lines
d	print date and time
v	print procedure name and version number
f	print file name being analyzed

Example:

```
lrc.header = "tld";
```

If *lrc*.`header` == “”, no header is printed. Default = “tldvf”.

lrc.`iter` scalar, sets the maximum number of iterations for the *iterative three-stage least squares regression.* The iterative process is also subject to the convergence criterion *lrc*.`tol`.

Default = 1.

lrc.`output` scalar, determines printing of intermediate results.

0 nothing is written.

1 serial ASCII output format suitable for disk files or printers.

Default = 1.

lrc.`pcor` scalar, if 1, print the correlation matrix of all coefficients in the system after convergence.
This is the $\left[Z'(\hat{\Sigma}^{-1} \otimes X(X'X)^{-1}X')Z\right]^{-1}$ matrix scaled to unit diagonals and is *not* the correlation matrix of variables.
Default = 0.

lrc.`pcov` scalar, if 1, print the covariance matrix of all coefficients in the system after convergence, which is
$\left[Z'(\hat{\Sigma}^{-1} \otimes X(X'X)^{-1}X')Z\right]^{-1}$. Default = 0.

lrc.`range` 2×1 vector, specifies the range of the data set to be used in estimation. The first element specifies the beginning observation while the second element specifies the ending observation. For example: *lrc*.`range`={ 100,200 }. Default is { 0,0 } and uses the whole data set.

lrc.`ranktol` scalar, specifies the tolerance used to determine if any of the singular values are effectively 0 when computing the rank of a matrix. Default = 10^{-13}.

lrc.`result` scalar.

1 print only Three-stage results.

2 print both Two-stage and Three-stage results.

Default = 2.

lrc.`row` scalar, specifies how many rows of the data set are read per iteration of the read loop. If *lrc*.`row` = 0, the number of rows to be read is calculated by the program. Default = 0.

lrc.`rowfac` scalar, "row factor". If an **LRMT** procedure fails due to insufficient memory while attempting to read a **GAUSS** data set, then *lrc*.`rowfac` may be set to some value between 0 and 1 to read a *proportion* of the original number of rows of the **GAUSS** data set. For example, setting

```
lrc.rowfac = 0.8;
```

causes **GAUSS** to read in only 80% of the rows that were originally calculated. This global has an effect only when *lrc*.`row` = 0. Default = 1.

lrc.`title` string, message printed at the top of the results. Default = "".

lrc.`tol` specifies a convergence criterion to stop the iterative process. The iterative process continues until either the iteration limit specified in *lrc*.`iter` is reached or the percentage change in the log of

determinant of $\hat{\Sigma}$ is less than the convergence criterion. Mathematically, the convergence criterion is written as follows:
$ABS\left[(\log\hat{\Sigma}_{current} - \log\hat{\Sigma}_{previous})/\log\hat{\Sigma}_{previous}\right] \times 100 \leq lrc.\texttt{tol}$
Default = 0.0001.

dataset string, name of **GAUSS** data set.

LHS_vars $M \times 1$ string array, all of the endogenous variables in the model.

RHS_vars $K \times 1$ string array, all of the right-hand side variable names in the systems. The order of the variable names must correspond to the order of the equations when they are stacked. For example:

```
X_vars = { const, x1, x2,..., xk,    /* equation 1 */
           const, y1, x1,..., xn,    /* equation 2 */
                      :
                      :
           const, y6, x2,..., xk };  /* equation M */
```

If a constant vector is desired for one particular equation, simply put "CONST" in the *RHS_vars* list.

EXO_vars $P \times 1$ string array, all of the exogenous variables in the system. Specify "CONST" in the *EXO_vars* list should a constant vector be desired.

Novars numeric vector to determine the number of right hand side variables in each equation. For example:

```
Novars = { 3, 4, 5 };
```

From the above, there are 3 right-hand side variables in the 1^{st} equation, 4 in the 2^{nd} equation, and 5 in the last equation.

Restrict string or 0, if *restrict* equals 0, estimation without restrictions is performed. Otherwise, the estimator is estimated with the given restrictions. The syntax for *Restrict* is as follows:

Restrict = "$restriction1, restriction2, \cdots, restrictionJ$";

More than one restriction is allowed provided each is separated by a comma. Each restriction must be written as a linear equation with all variables on the left hand side and the constant on the right hand side (i.e., $x1:1 + x1:2 = 1$). Variables shown in each restriction must be variables in the regression model. Note that the numeric value following the colon (:) signifies which equation the variable comes from (i.e., 3X4:10 indicates the X4 variable comes from the 10^{th} equation). Restrictions in the *Restrict* argument must be consistent and not redundant otherwise error messages occur. *Note that only the parameters associated with the variables are restricted and not the variables in the model.* For example:

```
restrict = "const:1 + const:2 + const:3 = 1,
            trend:1 = 0,trend:2 = 0,trend:3 = 0";
```

▪ Output

lro an instance of an **lrOut** structure, which contains all calculated statistics. The following members of *lro* are set by the **l3sls** routine.

lro.`model`	string, name of the estimation procedure.
lro.`nms`	$K \times 1$ string array, names of the regressors.
lro.`b`	$K \times 1$ vector, regression coefficients.
lro.`vc`	$K \times K$ matrix, variance-covariance matrix of the coefficients.
lro.`se`	$K \times 1$ vector, standard errors of the coefficients.
lro.`s2`	$M \times 1$ vector, variance of the estimate ($\hat{\sigma}^2$).
lro.`cx`	$K \times K$ matrix, correlation matrix of the coefficients.
lro.`rsq`	$M \times 1$ vector, R^2.
lro.`rbsq`	$M \times 1$ vector, adjusted R^2.
lro.`dw`	$M \times 1$ vector, Durbin-Watson statistic.
lro.`sse`	$M \times 1$ vector, residual sum of squares.
lro.`nobs`	scalar, number of observations.
lro.`ixtx`	$P \times P$ matrix, $(X'X)^{-1}$ of the $\left[Z'(\hat{\Sigma}^{-1} \otimes X(X'X)^{-1}X')Z\right]^{-1}$.
lro.`sigma`	$M \times M$ matrix, residual covariance matrix $\hat{\Sigma}$.
lro.`novars`	$M \times 1$ vector, number of RHS variables in each equation.
lro.`errcode`	scalar, zero or scalar error code.

If errors are encountered, they are handled with the low order bit of the trap flag.

trap 0	terminate with error message
trap 1	return scalar error code in *lro*.`errcode`

For more details on **trap**, see the *COMMAND REFERENCE* of the **GAUSS** manual. Since the returning error code appears as a missing value, it can be translated with the command **scalerr**(*lro*.`errcode`) or viewed with the **lrerror** procedure. See the **lrerror** procedure for more details. Definitions of the error codes can be found in Section 2.6.2 of this manual.

Remarks

The **l3sls** estimator and its asymptotic variance are as follows:

$$\hat{\delta}_{j,l3sls} = \left[Z'(\hat{\Sigma}^{-1} \otimes X(X'X)^{-1}X')Z \right]^{-1} \left[Z'(\hat{\Sigma}^{-1} \otimes X(X'X)^{-1}X')Y \right] \tag{1}$$

$$AVAR(\hat{\delta}_{j,l3sls}) = \left[Z'(\hat{\Sigma}^{-1} \otimes X(X'X)^{-1}X')Z \right]^{-1} \tag{2}$$

where X, Y and Z are the matrix of all exogenous variables, vector of endogenous variables, and matrix of all right-hand side variables, respectively, in the systems. $\hat{\Sigma}$ is the covariance matrix of the residuals and its elements $\hat{\sigma}_{ij}$ are computed from the *Two-stage least squares estimator*. That is:

$$\hat{\sigma}_{ij} = \frac{(Y_i - Z_i\hat{\delta}_{i,l2sls})'(Y_j - Z_j\hat{\delta}_{j,l2sls})}{T} \tag{3}$$

where T is the total number of observations.

Note:

1. Before using **l3sls**, you should remove all unidentified equations and all identities, since the latter have zero disturbances which would render the $\hat{\Sigma}$ matrix singular [7, page 490].

2. Although **l3sls** can perform Iterated Three-Stage Least Squares estimation, this does not guarantee a maximum likelihood estimator nor that the asymptotic efficiency will improve [6, page 633].

3. The default for the denominator in calculating the $\hat{\sigma}_{ij}$ is T. However, by changing *lrc*.`lrdv` to 0, you force the substitution of $(MT - K)/M$ for T as the divisor, where M is the number of equations, T is the number of observations, and K is the number of all estimated parameters in the model.

4. Linear hypothesis testing can be done with the **lrtest** procedure.

5. Restrictions on coefficients of different structural equations can be imposed. However, only linear restrictions are allowed.

6. R^2 calculated for each equation is not well defined and could be negative.

Missing data are handled automatically. That is, any observation which has a missing value for any variable is removed from computation.

Example

This example is from Judge, Hill, Griffiths, Lütkepohl, and Lee [8, page 656].

```
/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Program file: lrmt12.e
Data set:     tmt15_1
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/

library lrmt;
#include lrmt.sdf

struct lrControl lrc;
struct lrOut lro;
lrc = lrControlCreate;
output file = lrmt12.out reset;

dataset = "tmt15_1";
x = "const"$|"x2"$|"x3"$|"x4"$|"x5";
y = "y1"$|"y2"$|"y3";
string z = { "const","y2","y3","const","y1","x2",
             "x3","x4","const","y2","x2","x5" };
novars = { 3,5,4 };

lrc.result = 1;    /* Print only the three-stage results */
lrc.dv = 1;           /* Using N as divisor */
lro = l3sls(lrc,dataset,y,z,x,novars,0);
output off;

/*++++++  end of program file  ++++++++++++++++++++++++++++++++++++*/
```

Source

l3slsmt.src

Purpose

Sets the members of an **lrControl** structure to default values.

Library

lrmt

Format

lrc = **lrControlCreate;**

Output

lrc	an instance of an **lrControl** structure.

Remarks

It is generally good practice to put this instruction at the top of all command files that invoke procedures in the **LRMT** module. This prevents the members of *lrc* from being inappropriately defined when a command file is run several times or when a command file is run after another command file that calls **LRMT** procedures.

Source

`lrsetmt.src`

Purpose

lreg is a general procedure for linear regression. It applies the method of *Ordinary Least Squares* to perform multiple regression. References may be found in any standard textbook of statistics or econometrics.

Library

lrmt

Format

lro = **lreg(***lrc*,*dataset*,*depvar*,*indvars*,*Restrict***);**

Input

lrc an instance of an **lrControl** structure. The following members of *lrc* are referenced within the **lreg** routine:

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

t	print title (see *lrc*.`title`)
l	bracket title with lines
d	print date and time
v	print procedure name and version number
f	print file name being analyzed

Example:

```
lrc.header = "tld";
```

If *lrc*.`header` == “”, no header is printed. Default = “tldvf”.

lrc.`lregcol` scalar, if 1, perform collinearity diagnostics. Statistics calculated are described as above. Default = 0.

lrc.`lreghc` scalar, if 1, the heteroskedastic-consistent covariance matrix estimator is calculated. Default = 0.

lrc.`lregres` string, a file name to request influence diagnostics. Statistics generated from the diagnostics are saved under this file name. Besides the diagnostic statistics, the predicted values, dependent variable and independent variables are also saved. They are saved in the following order:

Col. #	*Name*	*Description*
1	RES	Residuals = (observed-predicted)
2	HAT	Hat Matrix Values
3	SRES	Standardized Residuals
4	RSTUDENT	Studentized Residuals
5	COOK	Cook Influence Statistics
6	YHAT	Predicted Values
7	*depname*	Dependent Variable
8 +	*indname*	Independent Variables

Default = “”.

lrc.`output` determines printing of intermediate results.

0 nothing is written.

1 serial ASCII output format suitable for disk files or printers.

Default = 1.

lrc.`pcor` scalar, if 1, print the correlation matrix of coefficients. This is the $(X'X)^{-1}$ matrix scaled to unit diagonals and is *not* the correlation matrix of variables. Default = 0.

lrc.`pcov` scalar, if 1, print the covariance matrix of coefficients, which is $\hat{\sigma}^2(X'X)^{-1}$, where $\hat{\sigma}^2$ is the mean squared error. Default = 0.

lrc.`range` 2×1 vector, specifies the range of the data set to be used in estimation. The first element specifies the beginning observation while the second element specifies the ending observation. For example: *lrc*.`range`={ 100,200 }. Default is { 0,0 } and uses the whole data set.

lrc.`ranktol` scalar, specifies the tolerance used to determine if any of the singular values are effectively 0 when computing the rank of a matrix. Default = 10^{-13}.

lrc.`row` scalar, specifies how many rows of the data set are read per iteration of the read loop. If *lrc*.`row` = 0, the number of rows to be read is calculated by the program. Default = 0.

lrc.`rowfac` scalar, “row factor”. If an **LRMT** procedure fails due to insufficient memory while attempting to read a **GAUSS** data set, then *lrc*.`rowfac` may be set to some value between 0 and 1 to read a *proportion* of the original number of rows of the **GAUSS** data set. For example, setting

```
lrc.rowfac = 0.8;
```

causes **GAUSS** to read in only 80% of the rows that were originally calculated. This global has an effect only when *lrc*.`row` = 0. Default = 1.

lrc.`title` string, message printed at the top of the results. Default = “”.

lrc.`weight` string, name of the weight variable. By default, unweighted least squares is calculated.

lrc.`weightindex` scalar, column number of weight variable. By default, unweighted least squares is calculated. *lrc*.`weightindex` overrides *lrc*.`weight`.

dataset string, name of **GAUSS** data set.

depvar	string, name of the dependent variable.
indvars	$K \times 1$ string array, all of the independent variable names. If a constant vector is desired, simply put "CONST" in the *indvars* list.
Restrict	string or 0, if *Restrict* equals 0, estimation without restrictions is performed. Otherwise, the *Restricted Least Squares* is done with the given restrictions. The syntax of *Restrict* is as follows:

Restrict $=$ "$restriction1, restriction2, \cdots, restrictionJ$";

More than one restriction is allowed provided each is separated by a comma. Each restriction must be written as a linear equation with all variables on the left hand side and the constant on the right hand side (e.g., *restrict* $=$ "$x1 + x2 = 1, x3 - x4 = 0$"). Variables shown in each restriction must be variables on the right-hand side of the equation. Restrictions in the *Restrict* argument must be consistent and not redundant otherwise error messages occur. *Note that only the parameters associated with the variables are restricted and not the variables.* For example:

```
restrict = "x11 + x22 + x33 = 1,
            x12 - x21 = 0, x23 - x32 = 0";
```

▪ Output

lro an instance of an **lrOut** structure, which contains all calculated statistics. The following members of *lro* are set by the **lreg** routine.

lro.`model`	string, name of the estimation procedure.
lro.`nms`	$K \times 1$ string array, names of the regressors.
lro.`b`	$K \times 1$ vector, regression coefficients.
lro.`hc`	$K \times K$ matrix, heteroskedastic-consistent covariance matrix of b, if requested in *lrc*.`lreghc`.
lro.`vc`	$K \times K$ matrix, variance-covariance matrix of the coefficients.
lro.`se`	$K \times 1$ vector, standard errors of the coefficients.
lro.`s2`	scalar, variance of the estimate ($\hat{\sigma}^2$).
lro.`cx`	$K \times K$ matrix, correlation matrix of the coefficients.
lro.`rsq`	scalar, R^2.
lro.`rbsq`	scalar, adjusted R^2.
lro.`dw`	scalar, Durbin-Watson statistic.
lro.`sse`	scalar, residual sum of squares.
lro.`sst`	scalar, total sum of squares.
lro.`nobs`	scalar, number of observations.

lro.`xtx` $K \times K$ matrix, moment matrix of X, (i.e., $X'X$).

lro.`xty` $K \times 1$ vector, the $X'Y$ matrix.

lro.`errcode` scalar, zero or scalar error code.

If errors are encountered, they are handled with the low order bit of the trap flag.

trap 0	terminate with error message
trap 1	return scalar error code in *lro*.`errcode`

For more details on **trap**, see the *COMMAND REFERENCE* of the **GAUSS** manual. Since the returning error code appears as a missing value, it can be translated with the command **scalerr**(*lro*.`errcode`) or viewed with the **lrerror** procedure. See the **lrerror** procedure for more details. Definitions of the error codes can be found in Section 2.6.2 of this manual.

Remarks

Some features of **lreg**:

- estimates parameters subject to linear constraints.
- performs *Weighted Least Squares*.
- calculates *Heteroskedastic-consistent Standard Errors*.
- performs both influence and collinearity diagnostics.

The *Ordinary Least Squares* estimator and its variances are as follows:

$$b_{ols} = (X'X)^{-1}X'Y$$
$$Var(b_{ols}) = \hat{\sigma}^2(X'X)^{-1}$$

where $\hat{\sigma}^2 = (Y - Xb)'(Y - Xb)/(T - K)$ and Y is the dependent variable, X is a list of independent variables. T and K are respectively the total number of observations and total number of estimated coefficients.

For estimated parameters subject to linear constraints, the restricted estimator and its variances are as follows:

$$b^* = b - (X'X)^{-1}R'\left[R(X'X)^{-1}R'\right]^{-1}(Rb - z)$$
$$Var(b^*) = \hat{\sigma}^2\left[(X'X)^{-1} - (X'X)^{-1}R'\left[R(X'X)^{-1}R'\right]^{-1}R(X'X)^{-1}\right]$$

where R and z are the restriction matrix and vector respectively. Both $\hat{\sigma}^2$ and X are already defined as above.

Weighted Least Squares

When the error variances are not equal, ordinary least squares estimations are unbiased and consistent *but not efficient*, (i.e., they are not the minimum variance estimates). Let the matrix W be a diagonal matrix containing the weights w_i and the weights be inversely proportional to the error variances (i.e., $Var(e_i) = \sigma^2/w_i$).

The *Weighted Least Squares* estimators are:

$$b_{wls} = (X'WX)^{-1}X'WY$$

Note that if $W = I$, as it would be for unweighted least squares, $b_{wls} = b_{ols}$. Weighted least squares is a special case of the generalized least squares. In calculating the weighted least squares estimator, the weights are chosen to be greater than zero and normalized to sum to the number of observations. To perform the weighted least squares estimation, you must assign the name of a weight variable to the **lrControl** structure member **weight** or the column number of a weight variable to the **lrControl** structure member **weightindex**.

Heteroskedastic-consistent Standard Errors

White [16] has demonstrated that in the absence of precise knowledge of the form of heteroskedasticity, it is still possible to obtain a consistent estimator of the covariance matrix of b. This *heteroskedasticity-consistent covariance matrix estimator* is defined as:

$$HC\ Var(b) = (X'X)^{-1}X'\hat{\Lambda}X(X'X)^{-1}$$

where $\hat{\Lambda}$ is a diagonal matrix holding all the squares of the errors (i.e., $diag(\hat{e}_1^2, \hat{e}_2^2, ..., \hat{e}_T^2)$). This estimator is extremely useful since the precise nature of the heteroskedasticity is not known most of the time. In order to calculate the $HC\ Var(b)$, you must set *lrc*.`lreghc` = 1.

Influence Diagnostics

Influence diagnostics provide the following statistics: hat or leverage values, standardized and studentized residuals, and Cook's distance measure. Let X_i and Y_i be the i^{th} observation of the X matrix and Y vector respectively, e_i the i^{th} residual (i.e., $e_i = Y_i - X_i b$), $\hat{\sigma}^2_{-i}$ the variance estimate of σ^2 without the i^{th} observation, and b_{-i} the least squares estimates after removing the i^{th} observation.

Hat or leverage values (h_i) are defined as follows:

$$h_i = X_i(X'X)^{-1}X_i'$$

h_i is a measure of how far the i^{th} observation is from the center of the data in terms of the X values. Thus, a large leverage value h_i indicates that the i^{th} observation is distant from the center of the X observations.

Standardized residuals: It can be shown that

$$Var(e_i) = \sigma^2(1 - h_i)$$

An unbiased estimator of this variance is:

$$Var(e_i) = \hat{\sigma}^2(1 - h_i)$$

The ratio of e_i to $\sqrt{Var(e_i)}$ is called the standardized residual (r_i):

$$r_i = \frac{ei}{\hat{\sigma}\sqrt{1 - h_i}}$$

Note that the residuals e_i have substantially different sampling variations if the leverage values h_i differ significantly. Hence, the advantage of r_i is that it has constant variance when the model is correct. Weisberg [15] refers to this as the internally studentized residual.

Studentized residuals (r_{-i}) are defined as:

$$r_{-i} = r_i\sqrt{\frac{\hat{\sigma}^2}{\hat{\sigma}^2_{-i}}}$$

The advantage of the studentized residual can be seen when the i^{th} observation is far from the center of the data. If the i^{th} observation is removed, $\hat{\sigma}^2_{-i}$ is smaller, which makes the studentized residual larger relative to the standardized residual.

Cook's distance measure (D_i) measures the change in the estimated coefficients when the i^{th} observation is removed from the regression. The larger the value of D_i, the greater is the change in the estimates.

$$D_i = \frac{(b_{-i} - b)'(X'X)(b_{-i} - b)}{(K - 1)\hat{\sigma}^2}$$

You must specify an output file name in the **lrControl** structure member **lregres** when requesting the influence diagnostics (e.g., *lrc*.`lregres`="filename").

Collinearity Diagnostics

There are a variety of ways in which to detect the presence of collinearity or multicollinearity. The following statistics or measures are provided:

- determinant of the correlation matrix of the regressors
- Theil's multicollinearity effect
- variance inflation factor and its tolerance
- eigenvalue of $X'X$
- condition number and proportion of variance of the estimate

See Judge, Hill, Griffiths, Lütkepohl, and Lee [8] for more details.

Determinant of the correlation matrix of the regressors: let $\mid R_{xx} \mid$ be the determinant of the correlation matrix of the independent variables. The value of the determinant declines with increasing collinearity. Its value ranges from 0 to 1. If the independent variables are orthogonal, the value is 1, whereas with perfect collinearity among the regressors, the value is zero.

Theil's multicollinearity effect (m): this measure has been suggested by Theil [14]. It is defined as:

$$m = \sum_h (R^2 - R_h^2)$$

where R^2 is the coefficient of determination when all of the variables are included in the regression and R_h^2 is the coefficient of determination when the X_h is excluded from the independent variable list. $R^2 - R_h^2$ is the incremental contribution due to X_h. If all of the regressors are orthogonal, m is the same as R^2. Deviations from R^2 indicate multicollinearity.

Variance Inflation Factor (VIF) and Tolerances (TOL): VIF measures the degree to which the variances of the estimated regression coefficients are inflated over linearly independent variables. The TOL is the reciprocal of the VIF (i.e., $TOL = 1/VIF$) and is compared with a tolerance limit (frequently used are 0.01, 0.001, or 0.0001), below which the variable is not entered into the model.

The VIF for the j^{th} variable is defined as:

$$VIF_j = \frac{1}{(1 - R_j^2)}$$

where R_j^2 is the coefficient of multiple determination when X_j is regressed on all of the remaining independent variables. For example, if there were four independent variables, R_3^2 would be the coefficient of determination from regressing X_3 on X_1, X_2, and X_4.

The largest VIF_j among all X variables is often used as an indicator of the severity of multicollinearity. A maximum VIF_j in excess of 10 indicates multicollinearity unduly influencing the least squares estimates. Note that if all variables are orthogonal to each other, both VIF and TOL are 1.

Eigenvalues and Condition Number: The eigenvalues are computed from the $X'X$ matrix. If one or more columns of X are linearly dependent, one or more of the eigenvalues is zero. When one or more columns are *nearly* linearly dependent, the ratio of the largest to the smallest eigenvalue is very large. The square root of this ratio is called the *condition number* (CN):

$$CN = \sqrt{\frac{\lambda_{max}}{\lambda_{min}}}$$

where λ_{max} and λ_{min} denote the maximum and minimum eigenvalues of $X'X$, respectively. Since the eigenvalues are dependent on the scaling of the data, it is better to normalize the data (i.e., $S(X'X)S$, where S is a diagonal matrix with $1/\sqrt{x_i'x_i}$ on the diagonals). If the regressors are orthogonal, CN is 1. Belsley, Kuh, and Welsch [1] suggest that a value of CN in excess of 30 indicates serious problems of collinearity.

For each variable, **lreg** prints both eigenvalues and a condition number along with the variance proportion of the estimates. Coefficients with proportions in excess of 0.5 may be regarded as seriously affected by the collinearity in the X matrix.

Note: For collinearity diagnostics, you must specify *lrc*.`lregcol`=1.

Missing data are handled automatically. That is, any observation which has a missing value for any variable is removed from computation.

Example

This example is from Judge, Hill, Griffiths, Lütkepohl, and Lee [8, page 871].

```
/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Program file: lrmt13.e
Data set:     tmt21_1
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/

library lrmt;
#include lrmt.sdf
```

```
struct lrControl lrc;
struct lrOut lro;
lrc = lrControlCreate;
output file = lrmt13.out reset;

dataset = "tmt21_1";
y = "c";
x = "const"$|"w"$|"p"$|"a";

lrc.lregcol=1;   /* Request collinearity diagnostics */

lro = lreg(lrc,dataset,y,x,0);

output off;

/*++++++  end of program file  +++++++++++++++++++++++++++++++++++*/
```

■ Source

lregmt.src

Purpose

Provides interpretation of the error code returned from the procedures. You may add comments in addition to the printed error message. In order to use this procedure, you must set the trap flag by putting **trap 1** in the command file.

Library

lrmt

Format

lrerror(*comment*,*errcode*);

Input

comment	string, user defined comment.
errcode	a scalar error code.

Example

In this example, a user traps the errors himself with the use of **trap**, **scalerr**, and **lrerror**.

```
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Program file: testerrmt2.e
Data set:     tmt15_1
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/

library lrmt;
#include lrmt.sdf

struct lrControl lrc;
struct lrOut lro1,lro2,lro3;
lrc = lrControlCreate;
output file = testerrmt2.out reset;

trap 1;                        /* Initialize the trap */
lrc.dv = 0;
x = "xx1"$|"x2"$|"x3"$|"x4"$|"x5"; /* User mistyped the x1 */

y = "y1";                      /* First equation */
z = "x1"$|"y2"$|"y3";
lro1 = l2sls(lrc,"tmt15_1",y,z,x,0);
if scalerr(lro1.errcode);
```

```
   lrerror("Error appears in the 1st equation",lro1.errcode);
   pause(3);
endif;

y = "y2";                     /* Second equation */
z = "x1"$|"y1"$|"x2"$|"x3"$|"x4";
lro2 = l2sls(lrc,"tmt15_1",y,z,x,0);
if scalerr(lro2.errcode);
   lrerror("Error appears in the 2nd equation",lro2.errcode);
   pause(3);
endif;

y = "y3";                     /* Third equation */
z = "x1"$|"y2"$|"x2"$|"x5";
lro3 = l2sls(lrc,"tmt15_1",y,z,x,0);
if scalerr(lro3.errcode);
   lrerror("Error appears in the 3rd equation",lro3.errcode);
   pause(3);
endif;

trap 0 ;                    /* Reset the trap */
output off;

/*+++++  end of program file  +++++++++++++++++++++++++++++++++++*/
```

■ Source

lrutilmt.src

Purpose

Performs linear hypothesis testing for all estimation modules in **LRMT**. For **lreg**, the hypothesis test calculates the F statistic, whereas for **l2sls**, **l3sls**, and **lsur**, the $Wald$ statistic is calculated. References can be found in Judge, Hill, Griffiths, Lütkepohl, and Lee [8], Judge, Griffiths, Hill, Lütkepohl, and Lee [9], Greene [6], and Johnston [7].

Library

lrmt

Format

stat = **lrtest(***lrc*,*lro*,*test***);**

Input

lrc an instance of an **lrControl** structure. The following members of *lrc* are referenced within the **lrtest** routine:

lrc.`lreghc` scalar, if 1, the heteroskedastic-consistent covariance matrix estimator is used. Default = 0.

lrc.`output` scalar, determines printing of intermediate results.

0 nothing is written.

1 serial ASCII output format suitable for disk files or printers.

Default = 1.

lrc.`ranktol` scalar, specifies the tolerance used to determine if any of the singular values are effectively 0 when computing the rank of a matrix. Default = 10^{-13}.

lro an instance of an **lrOut** structure, the results generated from the corresponding regression procedure.

test string that contains the set of linear restrictions to perform the hypothesis testing. *test* has the following form:

$test =$ "$test1, test2, \cdots, testJ$";

More than one *test* equation is allowed provided each is separated by a comma. Each *test* equation must be written as a linear equation with all variables on the left hand side and the constant on the right hand side. Variables shown in each equation must be variables in the regression model. Equations in the *test* argument must be consistent and not redundant, otherwise error messages are issued. For a system of simultaneous equations, every variable defined in each *test* equation must have a colon (:) and a numeric value following the variable. The numeric value indicates which equation the variable comes from (i.e., 3X4:10 indicates that the X4 variable comes from the 10^{th} equation). *Note that only the parameters associated with the variables are tested and not the variables in the model.* For example:

```
 /* for models such as lreg and l2sls */
test = "x11+x22+x33=1,x12-x21=0,x23-x32=0";

 /* for models such as l3sls and lsur */
test = "const:1 + const:2 + const:3 = 1,
        trend:1 = 0,
        trend:2 = 0,
        trend:3 = 0";
```

■ Output

stat scalar, test statistic for the corresponding regression.

If errors are encountered, they are handled with the low order bit of the trap flag.

trap 0 terminate with error message

trap 1 return scalar error code in *stat*

For more details on **trap**, see the *COMMAND REFERENCE* of the **GAUSS** manual. Since the returning error code appears as a missing value, it can be translated with the command **scalerr**(*stat*) or viewed with the **lrerror** procedure. See the **lrerror** procedure for more details. Definitions of the error codes can be found in Section 2.6.2 of this manual.

■ Remarks

For **lreg**, the F statistic is defined as follows:

$$F_{(J,T-K)} = \frac{(R\hat{\beta} - z)' \left[R(X'X)^{-1}R'\right]^{-1} (R\hat{\beta} - z)}{J\hat{\sigma}^2}$$

where J is the set of linear restrictions, T and K are the total number of observations and coefficients, respectively. R and z are the restriction matrix and vector respectively. X is the data matrix and $\hat{\sigma}^2$ is the estimated error variance.

For **l2sls**, **l3sls**, and **lsur**, the $Wald$ statistic is defined as:

$$\chi^2_{(J)} \;=\; (R\hat{\beta} - z')(R\hat{C}R')^{-1}(R\hat{\beta} - z)$$

where J is the set of linear restrictions, R and z are the restriction matrix and vector respectively. $\hat{\beta}$ is the corresponding estimated coefficients and $\hat{C}$ is the covariance matrix of $\hat{\beta}$.

Example

This example is from Judge, Hill, Griffiths, Lütkepohl, and Lee [8, page 460]. Data is already logged here.

```
/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Program file: lrmt14.e
Data set:     tmt11_3
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/

library lrmt;
#include lrmt.sdf

struct lrControl lrc;
struct lrOut lro;
lrc = lrControlCreate;
output file = lr14.out reset;

dataset = "tmt11_3";
lhs = "q1"$|"q2"$|"q3";
string rhs = { "const","p1","y",
        "const","p2","y",
        "const","p3","y" };
novars = { 3,3,3 };     /* No. of RHS variables in each eqn. */
test = "p1:1-p2:2=0,
        p1:1-p3:3=0";

lrc.dv=0;
lrc.output = 0;  /* Output of the call to lsur won't be printed */

  /*  lsur estimation without restriction imposed  */
lro = lsur(lrc,dataset,lhs,rhs,novars,0);

lrc.output = 1;   /* Print the result of lrtest */

 /*  Linear hypothesis testing by using lrtest  */
call lrtest(lrc,lro,test);

output off;

/*+++++  end of program file  +++++++++++++++++++++++++++++++++++*/
```

Output for the example:

```
-----  LSUR:  Results for Linear Hypothesis Testing  --------------

       Wald Chi-SQ(2) statistic =   1.138      Prob. =  0.566

-------------------------------------------------------------------
```

Source

lrtestmt.src

Purpose

lsur (*Linear Seemingly Unrelated Regression*) is a procedure for estimating a system of equations. It employs the technique of joint-generalized least squares, which uses the covariance matrix of residuals. Linear restrictions can be imposed on the coefficients within or across equations. Hypothesis testing for these linear restrictions can be tested with the **lrtest** procedure.

Library

lrmt

Format

lro = **lsur(***lrc*,*dataset*,*LHS_vars*,*RHS_vars*,*Novars*,*Restrict***);**

Input

lrc an instance of an **lrControl** structure. The following members of *lrc* are referenced within the **lsur** routine:

lrc.`dv` scalar, determines which divisor is used to compute the covariance matrix of residuals.

- **0** $T - (K/M)$ is used as divisor, where T is the number of observations, K is the number of all right hand side variables in the system, and M is the total number of equations. Hence, (K/M) is the average number of coefficients per equation.
- **1** T is used as divisor. Users are encouraged to use this, since it provides good asymptotic properties for the estimator.

Default = 1.

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

t	print title (see *lrc*.`title`)
l	bracket title with lines
d	print date and time
v	print procedure name and version number
f	print file name being analyzed

Example:

```
lrc.header = "tld";
```

If *lrc*.`header` == "", no header is printed. Default = "tldvf".

lrc.`iter` scalar, sets the maximum number of iterations for the *iterative seemingly unrelated regression*. Default = 1.

lrc.`output` scalar, determines printing of intermediate results.

0 nothing is written.

1 serial ASCII output format suitable for disk files or printers.

Default = 1.

lrc.`pcor` scalar, if 1, print the correlation matrix of all coefficients in the system after convergence. This is the $\left[X'\left(\hat{\Sigma}^{-1}\otimes I\right)X\right]^{-1}$ matrix scaled to unit diagonals and is *not* the correlation matrix of variables. Default = 0.

lrc.`pcov` scalar, if 1, print the covariance matrix of all coefficients in the system after convergence which is $\left[X'\left(\hat{\Sigma}^{-1}\otimes I\right)X\right]^{-1}$. Default = 0.

lrc.`range` 2×1 vector, specifies the range of the data set to be used in estimation. The first element specifies the beginning observation while the second element specifies the ending observation. For example: *lrc*.`range`={ 100,200 }. Default is { 0,0 } and uses the whole data set.

lrc.`ranktol` scalar, specifies the tolerance used to determine if any of the singular values are effectively 0 when computing the rank of a matrix. Default = 10^{-13}.

lrc.`row` scalar, specifies how many rows of the data set are read per iteration of the read loop. If *lrc*.`row` = 0, the number of rows to be read is calculated by the program. Default = 0.

lrc.`rowfac` scalar, "row factor". If an **LRMT** procedure fails due to insufficient memory while attempting to read a **GAUSS** data set, then *lrc*.`rowfac` may be set to some value between 0 and 1 to read a *proportion* of the original number of rows of the **GAUSS** data set. For example, setting

```
lrc.rowfac = 0.8;
```

causes **GAUSS** to read in only 80% of the rows that were originally calculated. This global has an effect only when *lrc*.`row` = 0. Default = 1.

lrc.`title` string, message printed at the top of the results. Default = "".

lrc.`tol` scalar, specifies a convergence criterion to stop the iterative process. The iterative process continues until either the iteration limit specified in *lrc*.`iter` is reached or the percentage change in the log of determinant of $\hat{\Sigma}$ is less than the convergence criterion. Mathematically, the convergence criterion is written as follows:

$ABS\left[(\log\hat{\Sigma}_{current}-\log\hat{\Sigma}_{previous})/\log\hat{\Sigma}_{previous}\right]\times 100\leq lrc.\texttt{tol}$

Default = 0.0001.

dataset string, name of **GAUSS** data set.

LHS_vars M×1 string array, all of the dependent variable names in the systems. For example:

```
lhs_vars = { y1, y2, y3,...., yM };
```

where y1 is the dependent variable for the 1st equation, y2 for the 2nd equation, and so on.

RHS_vars K×1 string array, all of the independent variable names in the systems. The order of the variable names must correspond to the order of the equations when they are stacked. For example:

```
rhs_vars = { const, x11, x12,..., x1k,    /* equation 1 */
             const, x21, x22,..., x2k,    /* equation 2 */
                          :
                          :
             const, xM1, xM2,..., xMk };  /* equation M */
```

If a constant vector is desired for a particular equation, simply put "CONST" in the *RHS_vars* list.

Novars M×1 numeric vector, to determine the number of right hand side variables in each equation. For example:

```
novars = { 3, 4, 5 };
```

From the above, there are 3 right hand side variables in the 1st equation, 4 in the 2nd equation, and 5 in the 3rd equation.

Restrict string or 0, if *restrict* equals 0, estimation without restrictions is performed. Otherwise, the estimator is estimated with the given restrictions. The syntax for *Restrict* is as follows:

$Restrict \ = \ \text{“}restriction1, restriction2, \cdots, restrictionJ\text{”};$

More than one restriction is allowed provided each is separated by a comma. Each restriction must be written as a linear equation with all variables on the left hand side and the constant on the right hand side (e.g., $x1 : 1 + x1 : 2 = 1$). Variables shown in each restriction must be variables in the regression model. Note that the numeric value following the colon (:) signifies which equation the variable comes from (i.e., 3X4:10 indicates the X4 variable comes from the 10th equation). Restrictions in the *Restrict* argument must be consistent and not redundant otherwise error messages are returned. *Note that only the parameters associated with the variables are restricted and not the variables in the model.*

For example, suppose one wants to estimate the following 2 equations:

$$S_{it} \ = \ \alpha_i \ + \ \sum_{j=1}^{2} \alpha_{ij} lnP_{jt} \ + \ \alpha_{iY} lnY_t \ + \ \gamma_i trend \ + \ \varepsilon_{it} \qquad i = 1, 2$$

and would like to impose three restrictions on it. For example, to impose the restrictions ($\alpha_1 + \alpha_2 = 1$) and ($\gamma_i = 0,\ \forall i$) on the model, we can write the following code:

```
library lrmt;
#include lrmt.sdf
struct lrControl lrc;
lrc = lrControlCreate;
dataset = "temp";
lhs = "s1"$|"s2";
string rhs = { "const","lp1","lp2","ly","trend",   /* equation 1 */
        "const","lp1","lp2","ly","trend" };   /* equation 2 */
novars = { 5,5 };
restrict = "const:1 + const:2 = 1,
            trend:1=0,
            trend:2=0";
call lsur(lrc,dataset,lhs,rhs,novars,restrict);
```

▪ Output

lro an instance of an **lrOut** structure, which contains all calculated statistics. The following members of *lro* are set by the **lsur** routine.

lro.`model`	string, name of the estimation procedure.
lro.`nms`	$K \times 1$ string array, names of the regressors.
lro.`b`	$K \times 1$ vector, regression coefficients.
lro.`vc`	$K \times K$ matrix, variance-covariance matrix of the coefficients.
lro.`se`	$K \times 1$ vector, standard errors of the coefficients.
lro.`s2`	$M \times 1$ vector, variance of the estimate ($\hat{\sigma}^2$).
lro.`cx`	$K \times K$ matrix, correlation matrix of the coefficients.
lro.`rsq`	$M \times 1$ vector, R^2.
lro.`rbsq`	$M \times 1$ vector, adjusted R^2.
lro.`dw`	$M \times 1$ vector, Durbin-Watson statistic.
lro.`sse`	$M \times 1$ vector, residual sum of squares.
lro.`nobs`	scalar, number of observations.
lro.`sigma`	$M \times M$ matrix, residual covariance matrix $\hat{\Sigma}$.
lro.`novars`	$M \times 1$ vector, number of RHS variables in each equation.
lro.`errcode`	scalar, zero or scalar error code.

If errors are encountered, they are handled with the low order bit of the trap flag.

trap 0	terminate with error message
trap 1	return scalar error code in *lro*.`errcode`

For more details on **trap**, see the *COMMAND REFERENCE* of the **GAUSS** manual. Since the returning error code appears as a missing value, it can be translated with the command **scalerr**(*lro*.`errcode`) or viewed with the **lrerror** procedure. See the **lrerror** procedure for more details. Definitions of the error codes can be found in Section 2.6.2 of this manual.

Remarks

A powerful feature in **lsur** is that it can perform *Iterative Seemingly Unrelated Regression*. The iterative process terminates when it meets the convergence criterion. Good references can be found in Judge, Hill, Griffiths, Lütkepohl, and Lee [8, Ch. 11], Judge, Griffiths, Hill, Lütkepohl, and Lee [9, Ch. 12], Greene [6, Ch. 17], and Johnston [7, Ch. 8].

The **lsur** estimator and variance are as follows:

$$\begin{aligned}\hat{\hat{\beta}}_{sur} &= \left[X'\left(\hat{\Sigma}^{-1}\otimes I\right)X\right]^{-1}\left[X'\left(\hat{\Sigma}^{-1}\otimes I\right)Y\right]\\ Var(\hat{\hat{\beta}}_{sur}) &= \left[X'\left(\hat{\Sigma}^{-1}\otimes I\right)X\right]^{-1}\end{aligned}$$

where X and Y are stacked by equations. $\hat{\Sigma}$ is the estimated covariance matrix of residuals from each equation and has elements given by

$$\hat{\sigma}_{ij} = \frac{\hat{e}_i\hat{e}_j}{T} = \frac{\sum_{t=1}^{T}\hat{e}_{it}\hat{e}_{jt}}{T} \qquad i,j = 1,2,....,M$$

where M and T stand for the number of equations and number of observations respectively.

For restrictions imposed on the coefficients, the restricted estimator and variance are as follows:

$$\begin{aligned}\hat{\hat{\beta}}^{*}_{sur} &= \hat{\hat{\beta}}_{sur} - \hat{C}R'\left(R\hat{C}R'\right)^{-1}\left(R\hat{\hat{\beta}}_{sur} - r\right)\\ Var(\hat{\hat{\beta}}^{*}_{sur}) &= \hat{C} - \hat{C}R'\left(R\hat{C}R'\right)^{-1}R\hat{C}\end{aligned}$$

where R and r are the restriction matrix and vector respectively. $\hat{C}$ is the covariance matrix of $\hat{\hat{\beta}}_{sur}$ and has the form $\left[X'\left(\hat{\Sigma}^{-1}\otimes I\right)X\right]^{-1}$.

Missing data are handled automatically. That is, any observation which has a missing value for any variable is removed from computation. In this case R^2 calculated for each equation is not well defined and could be negative.

Example

This example is from Judge, Hill, Griffiths, Lütkepohl, and Lee [8, page 460]. Data is already logged here.

```
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Program file: lrmt15.e
Data set:     tmt11_3
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/

library lrmt;
#include lrmt.sdf

struct lrControl lrc;
struct lrOut lro1,lro2;
lrc = lrControlCreate;
output file = lrmt15.out reset;

dataset = "tmt11_3";
lhs = "q1"$|"q2"$|"q3";
string rhs = { "const","p1","y",
        "const","p2","y",
        "const","p3","y" };
novars = { 3,3,3 };     /* Number of RHS variables in each eqn. */
restrict = "p1:1-p2:2=0,
            p1:1-p3:3=0";

lrc.dv = 0;   /* Using normal divisor */

  /*  lsur estimation without restriction imposed */
lro1 = lsur(lrc,dataset,lhs,rhs,novars,0);

 /*  lsur estimation with restriction imposed */
lro2 = lsur(lrc,dataset,lhs,rhs,novars,restrict);

format /rd 8,4;
print;
print "--------------------------------------------------";
print "The coeff. with restrictions imposed are as follows: ";
print "--------------------------------------------------";
fmt = { "%-10.8s", "%14.8f" };
```

```
answer = satostrC(rhs,fmt[1]) $~ ftostrC(lro2.b,fmt[2]);
answer = strcombine(answer,"",0);
print answer;

output off;

/*+++++  end of program file  ++++++++++++++++++++++++++++++++++++*/
```

Source

`lsurmt.src`

■ Purpose

Constructs the restriction matrix and the constant vector for single equation models. Both the restriction matrix and the constant vector can be used for linear hypothesis testing and restricted estimation.

■ Library

lrmt

■ Format

{ *R*,*z* } = **Rmatrix**(*lrc*,*Restrict*,*Varnames*);

■ Input

lrc an instance of an **lrControl** structure. The following member of *lrc* is referenced within the **Rmatrix** routine:

lrc.`ranktol` scalar, specifies the tolerance used to determine if any of the singular values are effectively 0 when computing the rank of a matrix. Default = 10^{-13}.

Restrict string, the restriction equations. The syntax of *Restrict* is as follows:

$Restrict\ =\ \text{“}restriction1, restriction2, \cdots, restrictionJ\text{”};$

More than one restriction is allowed provided each is separated by a comma. Each restriction must be written as a linear equation with all variables on the left hand side and the constant on the right hand side (i.e., $x1 + x2 = 1$). Variables shown in each restriction must be variables in the right-hand side of the equation. Restrictions in the *Restrict* argument must be consistent and not redundant otherwise error messages occur. Note that the corresponding variable names are used to represent the regression parameters.

Varnames $N \times 1$ string array, the variable names of the regression parameters.

■ Output

R matrix, the restriction matrix. If errors are encountered, they are handled with the low order bit of the trap flag.

trap 0 terminate with error message

trap 1 return scalar error code in *R*

For more details on **trap**, see the *COMMAND REFERENCE* of the **GAUSS** manual. Since the returning error code appears as a missing value, it can be translated with the command **scalerr**(R) or viewed with the **lrerror** procedure. See the **lrerror** procedure for more details. Definitions of the error codes can be found in Section 2.6.2 of this manual.

z the constant vector.

Example

Suppose you wish to perform a *Linear Hypothesis Testing*,

$$H_0: \quad R\beta = z$$

where the R matrix, β, and z vector are as follows:

$$R = \begin{bmatrix} 0 & 1 & -3 & 0 & 0 & -6 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad \beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \\ \beta_5 \\ \beta_6 \end{bmatrix}, \quad \text{and} \quad r = \begin{bmatrix} 0 \\ 2 \\ 0 \end{bmatrix}$$

That is to test jointly with the following equations:

$$\begin{aligned} \beta_1 - 3\beta_2 - 6\beta_5 &= 0 \\ \beta_3 + \beta_4 &= 2 \\ \beta_0 &= 0 \end{aligned}$$

By typing the following code, you can create the R matrix and z vector easily.

```
library lrmt;
#include lrmt.sdf

struct lrControl lrc;
lrc = lrControlCreate;

str = "x1-3x2-6x5=0, x3+x4=2, x0=0";
varnames = "x0"$|"x1"$|"x2"$|"x3"$|"x4"$|"x5"$|"x6";
{ R,z } = Rmatrix(lrc,str,varnames);
```

Source

rmatrixmt.src

Purpose

Constructs the restriction matrix and the constant vector for systems of equations models. Both the restriction matrix and the constant vector can be used for linear hypothesis testing and restricted estimation.

Library

lrmt

Format

{ *R*,*z* } = **SRmatrix(***lrc*,*Restrict*,*Varnames*,*Novars***);**

Input

lrc an instance of an **lrControl** structure. The following member of *lrc* is referenced within the **SRmatrix** routine:

lrc.`ranktol` scalar, specifies the tolerance used to determine if any of the singular values are effectively 0 when computing the rank of a matrix. Default = 10^{-13}.

Restrict string, the restriction equations. The syntax of *Restrict* is as follows:

$Restrict \; = \; \text{``}restriction1, restriction2, \cdots, restrictionJ\text{''};$

More than one restriction is allowed provided each is separated by a comma. Each restriction must be written as a linear equation with all variables on the left hand side and the constant on the right hand side (e.g., $x1:1+x1:2=1$). Variables shown in each restriction must be variables in the regression model. Note that the numeric value following the colon (**:**) signifies which equation the variable comes from (e.g., 3X4:10 indicates the X4 variable comes from the 10^{th} equation). Restrictions in the *Restrict* argument must be consistent and not redundant otherwise error messages occur. Note that the corresponding variable names in the model are used to represent the regression parameters.

Varnames $N \times 1$ string array, the variable names of the regression parameters.

Novars numeric vector to determine the number of right hand side variables in each equation. For example:

```
novars = { 3, 4, 5 };
```

From the above, there are 3 right hand side variables in the 1^{st} equation, 4 in the 2^{nd} equation, and 5 in the 3^{rd} equation.

Output

R	matrix, the restriction matrix. If errors are encountered, they are handled with the low order bit of the trap flag.

trap 0	terminate with error message
trap 1	return scalar error code in R

For more details on **trap**, see the *COMMAND REFERENCE* of the **GAUSS** manual. Since the returning error code appears as a missing value, it can be translated with the command **scalerr**(R) or viewed with the **lrerror** procedure. See the **lrerror** procedure for more details. Definitions of the error codes can be found in Section 2.6.2 of this manual.

z	the constant vector.

Example

Suppose one wants to perform a restricted estimation with the following model:

$$\begin{aligned} S_1 &= \alpha_1 + \alpha_{11}\ln P_1 + \alpha_{12}\ln P_2 + \varepsilon_1 \\ S_2 &= \alpha_2 + \alpha_{21}\ln P_1 + \alpha_{22}\ln P_2 + \varepsilon_2 \end{aligned}$$

In matrix notation, the above model is as follows:

$$\begin{bmatrix} S_1 \\ S_2 \end{bmatrix} = \begin{bmatrix} 1 & \ln P_1 & \ln P_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \ln P_1 & \ln P_2 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_{11} \\ \alpha_{12} \\ \alpha_2 \\ \alpha_{21} \\ \alpha_{22} \end{bmatrix} + \varepsilon$$

And the restrictions are

$$\begin{aligned} \alpha_{12} - \alpha_{21} &= 0 \\ \alpha 1 + \alpha 2 &= 1 \\ \alpha_{11} + \alpha_{12} &= 0 \\ \alpha_{21} + \alpha_{22} &= 0 \end{aligned}$$

Hence, for the restricted estimation to take place, we first have to construct the R matrix and the z vector.

$$R = \begin{bmatrix} 0 & 0 & 1 & 0 & -1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \quad \text{and} \quad z = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

With the use of **SRmatrix**, we can create the R and Z easily.

```
library lrmt;
#include lrmt.sdf

struct lrControl lrc;
lrc = lrControlCreate;

novars = { 3,3 };   /* Number of RHS variables in each equation */
str = "lnp2:1 - lnp1:2 = 0,
       const:1 + const:2 = 1,
       lnp1:1 + lnp2:1 = 0,
       lnp1:2 + lnp2:2 = 0";
varnames = "const"$|"lnp1"$|"lnp2"$|"const"$|"lnp1"$|"lnp2";
{ R,z } = SRmatrix(lrc,str,varnames,novars);
```

Source

`rmatrixmt.src`

Bibliography

[1] Belsley, D.A., E. Kuh, and R.E. Welsch. 1980. *Regression Diagnostics.* New York: John Wiley & Sons.

[2] Berndt E.R. 1991. *The Practice of Econometrics: Classic and Contemporary.* Addison-Wesley Publishing Company, Inc.

[3] Chambers, R.G. 1989. *Applied Production Analysis.* New York: Cambridge University Press.

[4] Chow, G.C. 1960. "Tests of Equality between Subsets of Coefficients in Two Linear Regressions." *Econometrica.* Vol. 28, pp.591-605.

[5] Fisher, F.M. 1970. "Tests of Equality between Sets of Coefficients in Two Linear Regressions: An Expository Note." *Econometrica.* Vol. 38, No. 2, pp.361-366.

[6] Greene, William H. 1990. *Econometric Analysis.* Macmillan.

[7] Johnston, J. 1984. *Econometric Methods.* 3rd Edition. Tokyo: McGraw-Hill International Book Co.

[8] Judge, G.G., R.C. Hill, W.E. Griffiths, H. Lütkepohl and T.C. Lee. 1988. *Introduction to the Theory and Practice of Econometrics.* 2nd Edition. New York: John Wiley & Sons.

[9] Judge, G.G., W.E. Griffiths, R.C. Hill, H. Lütkepohl and T.C. Lee. 1985. *The Theory and Practice of Econometrics.* 2nd Edition. New York: John Wiley & Sons.

[10] Klein, L.R. 1950. *Economic Fluctuations in the U.S., 1921–1941.* New York: John Wiley & Sons.

[11] Kmenta, J., and R.F. Gilbert. 1968. "Small Sample Properties of Alternative Estimators of Seemingly Unrelated Regressions." *Journal of American Statistical Association.* Vol. 63, pp.1180-1200, December.

[12] Maddala, G.S. 1989. *Introduction to Econometrics*. Macmillan.

[13] Silberberg, E. 1978. *The Structure of Economics: A Mathematical Analysis*. New York: Mcgraw Hill.

[14] Theil, H. 1971. *Principles of Econometrics*. New York: John Wiley & Sons.

[15] Weisberg, S. 1985. *Applied Linear Regression*. 2nd Edition. New York: John Wiley & Sons.

[16] White, H. 1980. "A Heteroskedasticity-Consistent Covariance Matrix and a Direct Test for Heteroskedasticity." *Econometrica*. Vol. 48, pp.817-838.

[17] Young, D.L., Mittelhammer, R.C., Rostamizadeh, A. and Holland, D.W. 1985. *Duality Theory and Applied Production Economics Research: A Pedagogical Treatise*. Agriculture Research Center, College of Agriculture and Home Economics, Washington State University. Research Bulletin No. 0962.

[18] Zellner, A. and H. Theil. 1962. "Three-Stage Least Squares: Simultaneous Estimation of Simultaneous Equations." *Econometrica*. Vol. 30, No. 1, pp.54-78.

[19] Zellner, A. 1962. "An Efficient Method of Estimating Seemingly Unrelated Regressions and Tests for Aggregation Bias." *Journal of the American Statistical Association*. Vol. 57, No. 3, pp.348-368.

Index