User’s Guide

by

William H. Greene
Econometric Software, Inc.
End-User License Agreement

This is a contract between you and Econometric Software, Inc. The *software product* refers to the computer software and documentation as well as any upgrades, modified versions, copies or supplements supplied by Econometric Software. By installing, downloading, accessing or otherwise using the software product, you agree to be bound by the terms and conditions of this agreement.

**Copyright, Trademark, and Intellectual Property**

This software product is copyrighted by, and all rights are reserved by Econometric Software, Inc. No part of this software product, either the software or the documentation, may be reproduced, distributed, downloaded, stored in a retrieval system, transmitted in any form or by any means, sold or transferred without prior written permission of Econometric Software. You may not modify, adapt, translate, or change the software product. You may not reverse engineer, decompile, dissemble, or otherwise attempt to discover the source code of the software product.

*LIMDEP*™ and *NLOGIT*™ are trademarks of Econometric Software, Inc. The software product is licensed, not sold. Your possession, installation and use of the software product does not transfer to you any title and intellectual property rights, nor does this license grant you any rights in connection with software product trademarks.

**Use of the Software Product**

You have only the non-exclusive right to use this software product. A single user license is registered to one specific individual, and is not intended for access by multiple users on one machine, or for installation on a network or in a computer laboratory. For a single user license only, the registered single user may install the software on a primary stand alone computer and one home or portable secondary computer for his or her exclusive use. However, the software may not be used on the primary computer by another person while the secondary computer is in use. For a multi-user site license, the specific terms of the site license agreement apply for scope of use and installation.

**Limited Warranty**

Econometric Software warrants that the software product will perform substantially in accordance with the documentation for a period of ninety (90) days from the date of the original purchase. To make a warranty claim, you must notify Econometric Software in writing within ninety (90) days from the date of the original purchase and return the defective software to Econometric Software. If the software does not perform substantially in accordance with the documentation, the entire liability and your exclusive remedy shall be limited to, at Econometric Software’s option, the replacement of the software product or refund of the license fee paid to Econometric Software for the software product. Proof of purchase from an authorized source is required. This limited warranty is void if failure of the software product has resulted from accident, abuse, or misapplication. Some states and jurisdictions do not allow limitations on the duration of an implied warranty, so the above limitation may not apply to you. To the extent permissible, any implied warranties on the software product are limited to ninety (90) days.
Econometric Software does not warrant the performance or results you may obtain by using the software product. To the maximum extent permitted by applicable law, Econometric Software disclaims all other warranties and conditions, either express or implied, including, but not limited to, implied warranties of merchantability, fitness for a particular purpose, title, and non-infringement with respect to the software product. This limited warranty gives you specific legal rights. You may have others, which vary from state to state and jurisdiction to jurisdiction.

Limitation of Liability

Under no circumstances will Econometric Software be liable to you or any other person for any indirect, special, incidental, or consequential damages whatsoever (including, without limitation, damages for loss of business profits, business interruption, computer failure or malfunction, loss of business information, or any other pecuniary loss) arising out of the use or inability to use the software product, even if Econometric Software has been advised of the possibility of such damages. In any case, Econometric Software’s entire liability under any provision of this agreement shall not exceed the amount paid to Econometric Software for the software product. Some states or jurisdictions do not allow the exclusion or limitation of liability for incidental or consequential damages, so the above limitation may not apply to you.
Preface to the Student Version of LIMDEP 9

LIMDEP is a general, integrated computer package for estimating the sorts of econometric models that are most frequently analyzed with cross section and panel data. Its range of capabilities include basic linear regression and descriptive statistics, the full set of techniques normally taught in the first year of an econometrics sequence, and a tremendous variety of advanced techniques such as nested logit models, parametric duration models, Poisson regressions with right censoring and nonlinear regressions estimated by instrumental variables and the generalized method of moments (GMM). LIMDEP’s menu of options is as wide as that of any other general purpose program available, though, as might be expected, longer in some dimensions and shorter in others. Among the signature features of LIMDEP is that you will find here many models and techniques that are not available in any other computer package. In addition to the estimation programs you need for your model building efforts, you will also find in LIMDEP all the analytical tools you need, including matrix algebra, a scientific calculator, data transformation features and programming language elements, to extend your estimators and create new ones.

This program has developed over many years (since 1980), initially to provide an easy to use tobit estimator - hence the name, ‘LIMted DEPendent variable models.’ With the current release of the program, it has spun off a major suite of routines for the estimation of discrete choice models. This program, NLOGIT, builds on the Nested LOGIT model. NLOGIT has now grown to a self standing superset of LIMDEP.

The student version of LIMDEP that is described in this manual is the full program. It is limited in the size of data set that you can analyze and the size of model that you can estimate. But, beyond those restrictions, the student version of LIMDEP is the full program. No features are restricted or disabled in any way.

To the best of our knowledge, the code of this program is correct as described. However, no warranty is expressed or implied. Users assume responsibility for the selection of this program to achieve their desired results and for the results obtained.

William H. Greene
Econometric Software, Inc.
15 Gloria Place
Plainview, New York 11803
January 2008
Preface
Preface to the User’s Guide for the Student Version of LIMDEP 9

This user’s guide is constructed specifically for the student who is using LIMDEP for the first time and is, most likely, taking their first course in econometrics. It is not simply a distillation of the full (2,000+ page) manual for LIMDEP. This guide is divided into two parts. Part I is a user’s guide to operation of the software. There will be several examples drawn largely from econometrics to illustrate the operation of the program. However, the purpose of this first part is to show you how to use the software, rather than to show you specifically how to estimate and analyze econometric models with the program. In Part I, we are concerned with features such as how to read a data set, how to transform data, and what you need to know about missing data. Part II is about econometrics. The various chapters in Part II follow essentially the sequence of topics that one might encounter in an econometrics course. Thus, in Part II, we will show you how to describe your data then how to use the linear regression model. This topic usually takes most of the first semester, and it occupies a large section of this part of the manual. We then proceed to some of the more advanced topics that would logically appear later in the course, such as two stage least squares (instrumental variable estimation) and basic discrete (binary) choice models. Part II ends with a brief survey of the many somewhat and extremely advanced features that are also available in LIMDEP, but not covered in a conventional econometrics course (or in this manual).

Having introduced the manual as above, we do emphasize, this user’s guide is not an econometrics or statistics text, and does not strive to be one. The material below will present only the essential background needed to illustrate the use of the program. In order to accommodate as many readers as possible, we have attempted to develop the material so that it is accessible to both undergraduates and graduate students. (For the latter, a text that would be useful to accompany this guide is Econometric Analysis, 6th Edition (William Greene, Prentice Hall, 2008), which was written by the author of both LIMDEP and this manual.)
# Student LIMDEP 9.0 Combined Table of Contents

## Part I  Reference Guide to Using LIMDEP

### Chapter 1  Introduction to LIMDEP
- 1.1 The LIMDEP Program  1-1
- 1.2 References for Econometric Methods  1-2

### Chapter 2  Getting Started
- 2.1 Introduction  2-1
- 2.2 Equipment  2-1
- 2.3 Installation  2-1
- 2.4 Registration  2-2
- 2.5 Execution – Beginning the LIMDEP Session  2-3
- 2.6 Components of a LIMDEP Session 2-4
- 2.7 Exiting LIMDEP and Saving Results  2-5
- 2.8 Fast Input of a Data Set with OPEN/LOAD  2-7
- 2.9 Starting LIMDEP from your Desktop or with a Web Browser  2-8

### Chapter 3  Operating LIMDEP
- 3.1 Introduction  3-1
- 3.2.1 Opening a Project  3-1
- 3.2.2 Opening an Editing Window  3-2
- 3.3 Using the Editing Window  3-4
- 3.4 A Short Tutorial  3-8
- 3.5 Help  3-14

### Chapter 4  LIMDEP Commands
- 4.1 Commands  4-1
- 4.2 Command Syntax  4-1
- 4.3 Naming Conventions and Reserved Names  4-2
- 4.4 Command Builders  4-4

### Chapter 5  Program Output
- 5.1 The Output Window  5-1
- 5.2 Editing Your Output  5-2
- 5.3 Exporting Your Output  5-2.

### Chapter 6  Application and Tutorial
- 6.1 Application – An Econometrics Problem Set  6-1
- 6.2 Assignment: The Linear Regression Model  6-1
- 6.3 Read the Raw Data  6-4
- 6.4 Tutorial Commands  6-9
- 6.5 Using LIMDEP for Linear Regression Analysis  6-12
  - 6.5.1 Obtain Descriptive Statistics  6-12
  - 6.5.2 Transformed Variables  6-13
  - 6.5.3 Saving and Retrieving Your Project  6-14
# Table of Contents

6.5.4 Time Series Plot of the Price Variables  6-16  
6.5.5 Simple Regression  6-19  
6.5.6 Multiple Regression  6-22  
6.5.7 Hypothesis Tests  6-25  
6.5.8 Test of Structural Break  6-28  

## Chapter 7 Essential of Data Management  
7.1 Introduction  7-1  
7.2 Reading and Entering Data  7-1  
7.2.1 The Data Area  7-1  
7.2.2 The Data Editor  7-1  
7.2.3 Reading Data Files Into LIMDEP  7-4  
7.2.4 General ASCII Files  7-6  
7.2.5 Reading a Spreadsheet File  7-7  
7.2.6 Missing Values in Data Files  7-8  
7.3 Computing Transformed Variables  7-9  
7.3.1 The CREATE Command  7-9  
7.3.2 Conditional Transformations  7-13  
7.3.3 Transformations Involving Missing Values  7-13  
7.3.4 CREATE Functions  7-14  
7.3.5 Expanding a Categorical Variable into a Set of Dummy Variables  7-17  
7.3.6 Random Number Generators  7-19  
7.4 Lists of Variables  7-21  
7.4.1 Lists of Variables in Model Commands  7-21  
7.4.2 Namelists  7-22  
7.4.3 Using Namelists  7-23  
7.5 The Current Sample of Observations  7-24  
7.5.1 Cross Section Data  7-25  
7.5.2 Time Series Data  7-28  
7.6 Missing Data  7-30  

## Chapter 8 Estimating Models  
8.1 Introduction  8-1  
8.2 Model Estimation Commands  8-1  
8.2.1 The Command Builder  8-3  
8.2.2 Output from Estimation Programs  8-7  
8.3 Model Components and Results  8-10  
8.3.1 Using Weights  8-10  
8.3.2 Model Output  8-11  
8.3.3 Retrievable Results  8-14  
8.3.4 Creating and Displaying Predictions and Residuals  8-16  

## Chapter 9 Using Matrix Algebra  
9.1 Introduction  9-1  
9.2 Entering MATRIX Commands  9-2  
9.2.1 The Matrix Calculator  9-2  
9.2.2 MATRIX Commands  9-4  
9.2.3 Matrix Output  9-5  
9.2.4 Matrix Results  9-6  
9.2.5 Matrix Statistical Output  9-7  
9.3 Using MATRIX Commands with Data  9-8
Table of Contents

9.3.1 Data Matrices 9-9
9.3.2 Computations Involving Data Matrices 9-10
9.4 Manipulating Matrices 9-11
9.4.1 Naming and Notational Conventions 9-11
9.4.2 Matrix Expressions 9-13
9.5 Entering, Moving, and Rearranging Matrices 9-15
9.6 Matrix Functions 9-17
9.7 Sums of Observations 9-19

Chapter 10 Scientific Calculator
10.1 Introduction 10-1
10.2 Command Input in CALCULATE 10-2
10.3 Results from CALCULATE 10-4
10.4 Forms of CALCULATE Commands – Conditional Commands 10-5
10.4.1 Reserved Names 10-6
10.4.2 Work Space for the Calculator 10-7
10.4.3 Compound Names for Scalars 10-7
10.5 Scalar Expressions 10-7
10.6 Calculator Functions 10-9
10.7 Correlation Coefficients 10-13

Chapter 11 Programming with Procedures
11.1 Introduction 11-1
11.2 The Text Editor 11-1
11.2.1 Placing Commands in the Editor 11-1
11.2.2 Executing the Commands in the Editor 11-2
11.3 Procedures 11-4
11.4 Defining and Executing Procedures 11-5
11.4.1 Executing a Procedure Silently 11-6
11.4.2 Parameters and Character Strings in Procedures 11-7

Part II Econometric Models and Statistical Analyses

Chapter 12 Econometric Model Estimation
12.1 Introduction 12-1
12.2 Econometric Models 12-1
12.3 Model Commands 12-1
12.4 Command Builders 12-3
12.5 Model Groups Supported by LIMDEP 12-3
12.6 Common Features of Most Models 12-5
12.6.1 Controlling Output from Model Commands 12-5
12.6.2 Robust Asymptotic Covariance Matrices 12-5
12.6.3 Predictions and Residuals 12-6

Chapter 13 Describing Data
13.1 Introduction 13-1
13.2 Summary Statistics 13-1
13.2.1 Weights 13-2
13.2.2 Missing Observations in Descriptive Statistics 13-2
13.2.3 Sample Quantiles 13-2
Table of Contents

13.3  Histograms  13-4
  13.3.1  Histograms for Continuous Data  13-4
  13.3.2  Histograms for Discrete Data  13-6
13.4  Cross Tabulations  13-6
13.5  Kernel Density Estimation  13-7
13.6  Scatter Plots and Plotting Data  13-10
  13.6.1  Printing and Exporting Figures  13-10
  13.6.2  Saving a Graph as a Graphics File  13-10
  13.6.3  The PLOT Command  13-12
  13.6.4  Plotting One Variable Against Another  13-12
  13.6.5  Plotting a Simple Linear Regression  13-13
  13.6.6  Time Series Plots  13-13
  13.6.7  Plotting Several Variables Against One Variable  13-14
  13.6.8  Options for Scaling and Labeling the Figure  13-15

Chapter 14  The Linear Regression Model
14.1  Introduction  14-1
14.2  Least Squares Regression  14-1
  14.2.1  Retrieveable Results  14-3
  14.2.2  Predictions and Residuals  14-3
  14.2.3  Robust Covariance Matrix Estimation  14-5
  14.2.4  Restricted Least Squares  14-5
  14.2.5  Hypothesis Tests in the Linear Model  14-7
14.3  Estimating Models with Heteroscedasticity  14-9
14.4  Correcting for First Order Autocorrelation  14-9
14.5  Two Stage Least Squares  14-11
  14.5.1  Robust Estimation of the 2SLS Covariance Matrix  14-11
  14.5.2  Model Output for the 2SLS Command  14-11
14.6  Panel Data Models  14-12
  14.6.1  Data Arrangement and Setup  14-12
  14.6.2  One Way Fixed and Random Effects Models  14-15
  14.6.3  One Way Fixed Effects Models  14-18
  14.6.4  One Way Random Effects Model  14-21

Chapter 15  Models for Discrete Choice
15.1  Introduction  15-1
15.2  Modeling Binary Choice  15-1
  15.2.1  Model Commands  15-2
  15.2.2  Output  15-2
  15.2.3  Analysis of Marginal Effects  15-6
  15.2.4  Robust Covariance Matrix Estimation  15-7
15.3  Ordered Choice Models  15-9
  15.3.1  Estimating Ordered Probability Models  15-10
  15.3.2  Model Structure and Data  15-10
  15.3.3  Output from the Ordered Probability Estimators  15-11
  15.3.4  Marginal Effects  15-13
Chapter 16  Censoring and Sample Selection
16.1 Introduction 16-1
16.2 Single Equation Tobit Regression Model 16-1
   16.2.1 Commands 16-2
   16.2.2 Results for the Tobit Model 16-3
   16.2.3 Marginal Effects 16-4
16.3 Sample Selection Model 16-4
   16.3.1 Regression Models with Sample Selection 16-5
   16.3.2 Two Step Estimation of the Standard Model 16-6

Index
This page intentionally left blank.
Part I Reference Guide

Chapter 1  Introduction to LIMDEP
Chapter 2  Getting Started
Chapter 3  Operating LIMDEP
Chapter 4  LIMDEP Commands
Chapter 5  Program Output
Chapter 6  Application and Tutorial
Chapter 7  Essential of Data Management
Chapter 8  Estimating Models
Chapter 9  Using Matrix Algebra
Chapter 10 Scientific Calculator
Chapter 11 Programming with Procedures
This page intentionally left blank.
1.1 The LIMDEP Program

LIMDEP is an integrated package for estimating and analyzing econometric models. It is primarily oriented toward cross section and panel data. But, many standard problems in time series analysis can be handled as well. LIMDEP’s basic procedures for data analysis include:

- descriptive statistics (means, standard deviations, minima, etc.), with stratification,
- multiple linear regression and stepwise regression,
- time series identification, autocorrelations and partial autocorrelations,
- cross tabulations, histograms, and scatter plots of several types.

You can also model many extensions of the linear regression model such as:

- heteroscedasticity with robust standard errors,
- autocorrelation with robust standard errors,
- multiplicative heteroscedasticity,
- groupwise heteroscedasticity and cross sectional correlation,
- the Box-Cox regression model,
- one and two way random and fixed effects models for balanced or unbalanced panel data
- distributed lag models, ARIMA, and ARMAX models,
- time series models with GARCH effects,
- dynamic linear models for panel data,
- nonlinear single and multiple equation regression models,
- seemingly unrelated linear and nonlinear regression models,
- simultaneous equations models.

LIMDEP is best known for its extensive menu of programs for estimating the parameters of nonlinear models for qualitative and limited dependent variables. (We take our name from LIMited DEPendent variables.) No other package supports a greater variety of nonlinear econometric models. Among LIMDEP’s more advanced features, each of which is invoked with a single command, are:

- univariate, bivariate and multivariate probit models, probit models with partial observability, selection, heteroscedasticity and random effects,
- Poisson and negative binomial models for count data, with fixed or random effects, sample selection, underreporting, and numerous other models of over and underdispersion,
- tobit and truncation models for censored and truncated data,
- models of sample selection with one or two selection criteria,
- parametric and semiparametric duration models with time varying covariates,
- stochastic frontier regression models,
- ordered probit and logit models, with censoring and sample selection,
- switching regression models,
Introduction to LIMDEP

- nonparametric and kernel density regression,
- fixed effects models, random parameters models and latent class models for over 25 different linear and nonlinear model classes,

and over fifty other model classes. Each of these allows a variety of different specifications. Most of the techniques in wide use are included. Among the aspects of this program which you will notice early on is that regardless of how advanced a technique is, the commands you use to request it are the same as those for the simplest regression.

LIMDEP also provides numerous programming tools, including an extensive matrix algebra package and a function optimization routine, so that you can specify your own likelihood functions and add new specifications to the list of models. All results are kept for later use. You can use the matrix program to compute test statistics for specification tests or to write your own estimation programs. The structure of LIMDEP’s matrix program is also especially well suited to the sorts of moment based specification tests suggested, for example, in Pagan and Vella (1989) – all the computations in this paper were done with LIMDEP. The programming tools, such as the editor, looping commands, data transformations, and facilities for creating ‘procedures’ consisting of groups of commands will also allow you to build your own applications for new models or for calculations such as complicated test statistics or covariance matrices.

Most of your work will involve analyzing data sets consisting of externally generated samples of observations on a number of variables. You can read the data, transform them in any way you like, for example, compute logarithms, lagged values, or many other functions, edit the data, and, of course, apply the estimation programs. You may also be interested in generating random (Monte Carlo) samples rather than analyzing ‘live’ data. LIMDEP contains random number generators for 15 discrete and continuous distributions including normal, truncated normal, Poisson, discrete or continuous uniform, binomial, logistic, Weibull, and others. A facility is also provided for random sampling or bootstrap sampling from any data set, whether internal or external, and for any estimation technique you have used, whether one of LIMDEP’s routines or your own estimator created with the programming tools. LIMDEP also provides a facility for bootstrapping panel data estimators, a feature not available in any other package.

1.2 References for Econometric Methods

This manual will document how to use LIMDEP for econometric analysis. There will be a number of examples and applications provided as part of the documentation. However, we will not be able to provide extensive background for the models and methods. A few of the main general textbooks currently in use are:

Chapter 2

Getting Started

2.1 Introduction

This chapter will describe how to install LIMDEP on your computer. Sections 2.5 to 2.7 describe how to start and exit LIMDEP.

2.2 Equipment

LIMDEP is written for use on Windows driven computers. As of this writing, we do not support operation on any Apple Macintosh computers. Windows emulation software that allows you to run Windows on Apple machines should allow LIMDEP to operate, but we are unable to offer any assurance, nor any specific advice.

2.3 Installation

To install LIMDEP, first close all applications. Insert the LIMDEP CD in your CD-ROM drive. The setup program should start automatically. If it does not, open My Computer or Windows Explorer (see Figure 2.2) and double click your CD-ROM drive to view the contents of the LIMDEP CD. To launch the Setup program, double click Setup. In either case, the installation wizard will start.

Installation proceeds as follows:

Step 1. Preparing to Install…: The InstallShield Wizard checks the operating system and configures the Windows installer. Installation proceeds to the next step automatically.

Step 2. Welcome to the InstallShield Wizard for LIMDEP 9.0: Note, LIMDEP is protected by copyright law and international treaties. Click Next to continue.

Step 3. Econometric Software End-User License Agreement: In order to install LIMDEP 9.0, you must select ‘I accept the terms in the license agreement.’ Click Next to continue.

Step 4. Welcome to LIMDEP 9.0: This window presents a brief introduction to LIMDEP 9.0. Please take a moment to read this information and click Next to continue.

Step 5. Destination Folder: This window indicates the destination folder for the program: C:\Program Files. (The complete folder location is C:\Program Files\Econometric Software\LIMDEP9\Program.) Click Next to continue.

Step 6. Ready to Install the Program: This window reviews the installation instructions. Click Install to install LIMDEP 9.0. Installation takes about 60 seconds.

Step 7. InstallShield Wizard Completed: Click Finish to complete installation. (You do not have to restart your computer to complete installation.)

Step 8. The first time you launch LIMDEP, you will be presented with the Welcome and Registration dialog box. See Section 2.3.1 for a complete explanation of the registration process.
Installation also creates a resource folder, C:\LIMDEP9 with three subfolders:

- C:\LIMDEP9\Data Files
- C:\LIMDEP9\LIMDEP Command Files
- C:\LIMDEP9\Project Files

**NOTE:** All sample data files referenced in this documentation will be found in these folders.

### 2.4 Registration

The first time you use *LIMDEP* you will be presented with the **Welcome and Registration** dialog box. There are two steps to register *LIMDEP*. First, provide the registration information requested in the dialog box. Carefully input the serial number included with your program. This will place the registration information, including your serial number, in the **About** box. You must complete all three fields of this dialog box in order to begin using *LIMDEP*. See Figure 2.1.

Second, send your registration information to Econometric Software. You can register with Econometric Software by completing the registration card included with your order and faxing or mailing it to us. You can also send your registration information to Econometric Software online via our website, [www.limdep.com](http://www.limdep.com). To submit your registration information on our website, click **Help**, then select **LIMDEP Web Site** and proceed to the Registration page.

![Figure 2.1 Welcome and Registration Dialog Box](image-url)
2.5 Execution – Beginning the LIMDEP Session

Start LIMDEP as you would any other program, for example by double clicking the LIMDEP icon on your desktop.

The opening window is the LIMDEP desktop, shown in Figure 2.2. At the top of the screen, the main menu is shown above the LIMDEP toolbar. Below the toolbar is the command bar discussed in Chapter 3. The open window is the project window. The session is identified as your ‘project,’ which will ultimately consist of your data and the various results that you accumulate. This is where you will begin your LIMDEP session. Operation is discussed in Chapter 3.

Figure 2.2 LIMDEP Desktop Window

If you do not see the command bar when you first start the program, so that your initial desktop appears as below in Figure 2.3, then select Tools in the desktop menu and Options... from the drop down menu. In the dialog that appears next, as shown in Figure 2.4, tick the option to “Display Command Bar” to change the desktop menu so that it will then appear as in Figure 2.2, with the command bar included.
2.6 Components of a *LIMDEP* Session

When you are operating *LIMDEP*, you are accumulating a project that consists of at least four components:

- Your data, matrices, scalars, the environment, and so on. The window associated with this information is the project window – usually at the upper left of your screen as in Figure 2.2..
- The commands that you have accumulated on the screen in an editing window.
- The output that you have accumulated in the output window.
- *LIMDEP*’s session trace file described in Section 2.11.

Figure 2.5 shows an example. In this session, we are analyzing a data set that contains 840 observations on about 20 variables. The chapters to follow will describe the various components and how the analysis proceeds. Figure 2.5 shows a fairly typical arrangement of a *LIMDEP* session. (The screen parts are arranged for the figure. It will be more conveniently spaced when you use the program.) The four parts of the session can be seen in the figure:

- The project consists of the CLOGIT data, which we will use later in several examples.
- The editing window (also referred to as a text or command editor) at the upper right shows the one command that we have entered.
- The output window (mostly obscured) is in the lower half of the split window.
The trace of the session is shown in the center of the screen, at the top of the split output window.

Figure 2.5 *LIMDEP* Data Analysis Session

### 2.7 Exiting *LIMDEP* and Saving Results

To leave *LIMDEP*, select Exit from the File menu. Whenever you exit a session, you should save your work. At any time in any session, you can save all of *LIMDEP*’s active memory, tables, data matrices, etc. into a file, and retrieve that file later to resume the session.

When you exit, *LIMDEP* will ask if you wish to save the contents of the editing, project and output windows. In each case, you may save the component as a named file. The query in each case is

```
! Save changes to ...<name>...
```

where `<name>` is the name that appears in the title banner of each of the active windows. See Figure 2.6 for an example, where the query refers to the output window. You may also have other
Getting Started

working windows open, such as graphs or your scientific calculator and, if so, you can save these as well. This operation is discussed further in Chapter 3.

Figure 2.6 Exiting LIMDEP – Saving Window Contents

The filename extension for a saved project window is .LPJ. The extension for a saved editing window file or output window is .LIM. When you use LIMDEP’s dialog box to save the project, editing or output windows, LIMDEP will remember the name of the file. When you return, you will be able to select the file from those listed in the File menu. The files listed 1-4 are the last four editing or output window files saved by LIMDEP, and the files listed 5-8 are the last four project files. (See Figure 2.7.) Just click the file name in the File menu to open the file. You can also save files by using the save options in the File menu.

NOTE: A saved editing window is referred to as a command or input file.

WARNING: Output files and command files are both saved with the .LIM extension. You will need to make careful note of which files you save are which type.
2.8 Fast Input of a Data Set with OPEN/LOAD

File:Save (or just saving the files upon Exit) then File:Open offers an extremely fast and easy way for you to transform ASCII or any other raw format data to binary format for quick input. The first time you use the data set, use File:Save or the LIMDEP SAVE command – see Chapter 3 – to rewrite it in LIMDEP’s binary format. (You can create any new variables you want to before doing so, these will be written with the raw data.) Later, instead of reading the raw data set, just use File:Open or any of the Windows Explorers to retrieve the file you saved earlier. If you created any new variables before you saved the data set, they will be loaded as well. The time savings in reading an ASCII data set which is unformatted are about 90%! For a formatted data set, it is still at least three times as fast.

TIP: The popular data transfer program Stat/Transfer supports LIMDEP and will translate many common system files, such as SAS, Stata, SYSTAT, TSP, EViews, and so on to .LPJ style LIMDEP project files.
2.9 Starting *LIMDEP* from your Desktop or with a Web Browser

Files with the .LPJ and .LIM extensions are ‘registered’ in your Windows system setup. This means that whenever you double click any file with a .LPJ or .LIM extension in any context, such as Find File, My Computer, any miniexplorer, or in a web browser, Windows will launch *LIMDEP*, and open the file. Note, however, in order to operate *LIMDEP*, you must have a project file open, not just a command or input file.

For example, you can create shortcuts by moving any .LIM or .LPJ files you wish to your Windows desktop. These files will then appear as icons on your desktop, and you can launch *LIMDEP* from your desktop. Similarly, if you open a .LIM file on our website or someone else’s, or a .LIM file that is sent to you as an email attachment, you can launch *LIMDEP* and place the indicated file in an editing window. If you start *LIMDEP* in this way, you must then use File:New/Project to open a new project window.

**NOTE:** Until you open a project, no other program functions are available. You must now open a project with any of the options in the File menu in order to proceed.
Chapter 3
Operating LIMDEP

3.1 Introduction

This chapter will explain how to give commands to LIMDEP and will describe some essential features of operation. The sections to follow are:

3.2: Beginning the LIMDEP Session
3.3: Using the Editing Window
3.4: A Short Tutorial
3.5: Help

3.2 Beginning the LIMDEP Session

When you begin your LIMDEP session from the Start:Programs menu or a shortcut on your desktop, the initial screen will show a project window entitled ‘Untitled Project 1’ and an empty desktop as shown in Figure 3.1. You can now begin your session by starting a new project or reloading an existing one. Figure 3.2 shows the File menu (the lower sections show some of our previous work).

3.2.1 Opening a Project

You can select Open or Open Project in the File menu (they are the same at this point) to reload a project that you saved earlier or select one of the existing projects known to LIMDEP (if any are). You may also select New to begin a new project.

**NOTE:** In order to operate LIMDEP, you must have a project open. This may be the default untitled project or a project that you created earlier. You will know that a project is open by the appearance of a project window on your desktop. Most of LIMDEP’s functions will not operate if you do not have a project open.

Note that the window shows that the data area has 3,333 rows. Your project window will show a different value. This is determined by a setting of the size of the data area.

**TIP:** You can associate a LIMDEP project file (see Section 2.9) with the program, and launch LIMDEP directly with your project file. Use My Computer or the Windows Explorer to navigate to the folder where you have created your project file – its name will be <the name>.LPJ. Drag the icon for the .LPJ file to your desktop, then close My Computer. Now, you can double click the icon on the desktop to launch LIMDEP and open the project file at the same time to begin your session.
3.2.2 Opening an Editing Window

To begin entering commands you should now open an editing window. Select New in the File menu to open the File:New dialog box as shown in Figure 3.3. Now, select Text/Command Document and OK to open the editing window, which will appear to the right of the project window.

**TIP:** You can press Ctrl-N at any time to bring up the File:New dialog box.

The desktop will now appear as shown in Figure 3.4, and you can begin to enter your commands in the editing window as we have done in an example in the figure. (We have arranged the various windows for appearance in our figures. Your desktop will be more conveniently arranged, and will be full screen sized.)
NOTE: If you have created a text file that contains *LIMDEP* commands that you will be using, instead of creating a new set of commands, you can use File:Open to open that file. If the file that you open has a .LIM file extension in its filename, then *LIMDEP* will automatically open an editing window and place the contents of the file in the window. You can also use Insert:Text File to place a copy of a text file (any text file) in an editing window that is already open.
Figure 3.4  Project Window and Editing Window

Note that the editing window shown in Figure 3.4 is labeled ‘Untitled 1.’ This means that the contents of this window are not associated with a file; the commands in an untitled window are just added to the window during the session. When you open a ‘.LIM’ file, the file will be associated with the window, and its name will appear in the window banner. The ‘*’ in the title means that the contents of this window have not yet been saved.

**TIP:** You can also associate a LIMDEP command file, <the name>.LIM, with LIMDEP. As in the earlier tip, if you use My Computer and your mouse to drag the icon for a .LIM file to the desktop, then you can double click the icon to start LIMDEP and at the same time, open an editing window for this command file. Note, however, that when you do this, you must then either open an existing project file with File:Open, File:Open Project, or one of the menu entries, or start a new project with File:New/Project/OK.

### 3.3 Using the Editing Window

LIMDEP’s editing window is a standard text editor. Enter text as you would in any other Windows based text editor. The Edit menu provides standard Undo, Cut, Copy, Paste, Clear, Select All, Find, Replace, and so on, as shown in Figure 3.5. You can also use the Windows clipboard functions to move text from other programs into this window, or from this window to your other programs. You can, for example, copy text from any word processor, such as Microsoft Word, and paste it into the editing window. The LIMDEP editing window will inherit all the features in
your word processor, including fonts, sizes, boldface and italic, colors, math objects, etc. However, once you save, then retrieve this window, these features will be lost, and all that will remain will be the text characters, in Courier font.

Figure 3.5  Editing Window and the Edit Menu

**TIP:** The text editor uses a Courier, size 9 font. If you are displaying information to an audience or are preparing materials for presentation, you might want to have a larger or different font in this window. You can select the font for the editor by using the Tools:Options/Editor:Choose Font menu. You may then choose a different font and size for your displays. This font will be used in the text editing window, and in the output window.

**NOTE:** The editing window is also referred to as a text editor or a command editor.
When you are ready to execute commands, highlight the ones you wish to submit with your mouse by placing the cursor at the beginning of the first line you wish to submit and, while holding down the left mouse button, moving the mouse cursor to the end of the last line you wish to submit. (This is the same movement that you use in your word processor to highlight text.) The highlighted section will change from black text on white to white text on black. Then, to execute the commands you may do either of the following:

- Click GO on the LIMDEP toolbar. (See below. If the toolbar is not showing on your screen, select the Tools:Options/View tab, then turn on the Display Tool Bar option.) The GO button is seen in green in the tool bar in Figure 3.6.

- Select the Run menu at the top of your screen. The Run menu is shown in Figure 3.7. The first two items in this menu are:
  
  - Run Line (or Run Selection if multiple lines are highlighted) will allow you to execute the selected commands once.
  - Run Line Multiple Times (or Run Selection Multiple Times if multiple lines are highlighted) will allow you to specify that the selected commands are to be executed more than one time. The dialog box queries you for the number of times.

The commands you have selected will now be carried out. In most cases, this will produce some output. LIMDEP will now automatically open a third window, your output window, discussed in Section 3.4.
If your commands fit on a single line – many of LIMDEP’s commands do not, there are additional ways for you to submit commands:

- You can type a one line command in the command window, then press Enter to submit it. The command window or command bar is the small window located below the LIMDEP toolbar. The ▼ button at the end of the line allows you to recall and select among the last several such commands you have submitted. See Figure 3.6.

- You can submit a single line of text in the text editor to the command processor just by placing the cursor anywhere on that line (beginning, middle or end), and then clicking the GO button. The single line does not have to be highlighted for this.

- For desktop computers (this does not apply to most laptop or notebook computers), the two Enter keys on your keyboard, one in the alphabetic area and one in the numeric keypad, are different from LIMDEP’s viewpoint. You can submit the line with the cursor in it as a one line command by pressing the numeric Enter key. The alphabetic Enter key acts like an ordinary editing key.

Finally, the right mouse button is also active in the editing window. The right mouse button invokes a small menu that combines parts of the Edit and Insert menus, as shown in Figure 3.8. As in the Edit menu, some entries (Cut, Copy) are only active when you have selected text, while Paste is only active if you have placed something on the clipboard with a previous Cut or Copy. Run Line is another option in this menu. Run Line changes to Run Selection when one or more lines are highlighted in the editing window, and if you make this selection, those lines will be submitted to the program. If no lines are highlighted, this option is Run Line, for the line which currently contains the cursor.
You may have noted by this point, that operating of LIMDEP uses a mixture of menus and commands submitted from the text editor. Though most modern software is largely “point and click” menu driven, much of what you do in econometrics is not very well suited to this style of processing. In fact, it is possible to operate LIMDEP almost without using your keyboard once your data are entered and ready. We will demonstrate use of the menus for model building as we go along. But, you will almost surely find that using the command editor and the LIMDEP command language is far simpler and more efficient than using the menus and dialog windows. Moreover, when you do begin to write your own computations, such as using matrix algebra where you must compose mathematical expressions, then the menus will no longer be useful.

3.4 A Short Tutorial

Start the Program.

We assume that you have successfully installed LIMDEP on your computer and created a shortcut to use to invoke the program. Now, invoke LIMDEP, for example, by double clicking the shortcut icon or from the Start:Programs menu. The desktop will appear as shown in Figure 3.9, with a new project window open, and no other windows active.
Open an Editing Window.

Select File:New, then Text/Command Document to open an editing window, exactly as discussed in Section 3.2.2. (See Figure 3.3.)

Place Commands in the Editing Window.

Type the commands shown in the editing window of Figure 3.10. These commands will do the following:

1. Instruct LIMDEP to base what follows on 100 observations.
2. Create two samples of random draws from the normal distribution, a ‘y’ and an ‘x.’
3. Compute the linear regression of y on x.

Spacing and capitalization do not matter – type these three lines in any manner you find convenient. But, do use three lines. (You need not type the comment lines “I will use ... data.”
Submit the First Two Commands.

Highlight the first two lines of this command set. Now, move the mouse cursor up to the (now) green button marked GO – it is directly below Tools – and click the GO button. Note that a new window appears, your output window – see Figure 3.11. (You may have to resize it to view the output.)

The output window will always contain a transcript of your commands. Since you have not generated any numerical results, at this point, that is all it contains.
Compute the Regression.

Before doing this step, notice that the top half of the output window has the Trace tab selected. If you click the Status tab, this will change the appearance of the top half of the window, as you’ll see later. The trace feature in the output window is useful when you execute iterative, complicated nonlinear procedures that involve time consuming calculations. The status window will help you to see how the computation is progressing, and if it is near completion.

Now, select the last line in your command set, the REGRESS command, and click the GO button. The regression output appears in the lower half of the window, and you can observe the accumulating trace in the upper half of the window. This trace in the top half of the window will be recorded as the trace file, TRACE.LIM when you exit the program. (Exit status of 0.0 means that the model was estimated successfully.)
Figure 3.12 Regression Output in Output Window

The Project Window.

Note in Figure 3.9, in the project window, that the topics Matrices and Scalars have symbols next to them, indicating that the topic can be ‘expanded’ to display its contents. But, the Variables entry is not marked. After you executed your second line in your editing window, and created the two variables $x$ and $y$, the Variables topic is now marked with . Click this symbol to expand the topic. The REGRESS command created another variable, logl_obs. It also created three matrices, as can be seen in Figure 3.13.
Some other features you might explore in the project window:

- Click the symbol next to the Matrices and/or Scalars topics.
- Double click any name that you find in the project window in any of the three topics.
- Single click any of the matrix or scalar names, and note what appears at the bottom of the window.

Exit the Program.

When you are done exploring, select File: Exit. There is no need to save any of these windows, so answer no to the three queries about saving your results.
3.5 Help

NOTE: LIMDEP’s help feature uses a .hlp format file. This is (was) a standard Windows format feature, and its operation relies on a Windows utility program. With the release of Windows Vista, without notification, Microsoft made a decision to end support of this format, and did not include a version of the help engine in the new operating system. Thus, the Help procedure in LIMDEP (and many other programs) is not compatible with Windows Vista. But, evidently in response to the predictable protest, Microsoft has produced a patch for this feature that can be added to the operating system to revive help engines such as ours. Please go to http://www.limdep.com/support/faq/ for more information about this as well as instructions on how to obtain the patch.

LIMDEP offers an extensive Help file. Select Help:Help Topics from the menu to bring up the help editor. LIMDEP’s Help file is divided into seven parts, or ‘books.’ In the first book, you will find a selection of Topics that discuss general aspects of operating the program. The second book is the Commands list. This contains a list of the essential features and parts of all of LIMDEP’s commands. The third book contains a summary of the various parts of the desktop. The fourth book contains descriptions of updates to LIMDEP that were added after the manual went to press. Finally, there are three books of useful ancillary material: a collection of LIMDEP programs, some of which appear in the manual for the program, a collection of data sets that can be used for learning how to use LIMDEP and for illustrating the applications – these include the data sets used in the applications in this manual, and, finally, some of the National Institute of Standards accuracy benchmark data sets. The files in the last three books are also available in a resource folder created when LIMDEP is installed. The location for the folder is C:\LIMDEP9, and there are three subfolders, Data Files, LIMDEP Command Files, and Project Files.
Chapter 4

LIMDEP Commands

4.1 Commands

There are numerous menus and dialog boxes provided for giving instructions to LIMDEP. But, ultimately, the large majority of the instructions you give to the program will be given by commands that you enter in the text editor. This chapter will describe the LIMDEP command language. We begin by describing the general form and characteristics of LIMDEP commands. Section 4.4 will illustrate how the menus and dialog boxes can also be used to operate the program.

4.2 Command Syntax

All program instructions are of the form:

VERB ; specification ; specification ; ... ; specification $

The verb is a unique four character name which identifies the function you want to perform or the model you wish to fit. If the command requires additional information, the necessary data are given in one or more fields separated by semicolons (;). Commands always end with a ‘$’. The set of commands in LIMDEP consists generally of data setup commands such as reading a data file, data manipulation commands such as transforming a variable, programming commands such as matrix manipulation and scientific calculation commands, and model estimation commands. All are structured with this format. Examples of the four groupings noted are:

**READ** ; File = “C:\WORK\FRONTIER.DAT” ; Nobs = 27 ; Nvar = 4 $
**CREATE** ; logq = Log(output) $
**MATRIX** ; identity = Iden(5) ; bols = <X’X> * X’y $
**REGRESS** ; Lhs = logq ; Rhs = one, Log(k), Log(l) ; Plot residuals $

The following command characteristics apply:

- You may use upper or lower case letters anywhere in any command. All commands are translated to upper case immediately upon being read by the program, so which you use never matters. But, note that this implies that you cannot use upper and lower case as if they were different in any respect. That is, the variable CAPITAL is the same as capital.

- You may put spaces anywhere in any command. LIMDEP will ignore all spaces and tabs in any command.

- Every command must begin on a new line in your text editor.
• In any command, the specifications may always be given in any order. Thus,

\begin{verbatim}
READ ; Nobs = 100 ; File = DATA.PRJ $, and
READ ; File = DATA.PRJ ; Nobs = 100 $
\end{verbatim}

are exactly the same.

• You may use as many lines as you wish to enter a command. Just press Enter when it is convenient. Blank lines in an input file are also ignored.

• Most of your commands will fit on a single line. However, if a command is particularly long, you may break it at any point you want by pressing Enter. The ends of all commands are indicated by a $$. LIMDEP scans each line when it is entered. If the line contains a $,$ the command is assumed to be complete.

**HINT:** Since commands must generally end with a $$, if you forget the ending $$ in a command, it will not be carried out. Thus, if you submit a command from the editor and ‘nothing happens,’ check to see if you have omitted the ending $$ on the command you have submitted. Another problem can arise if you submit more than one command, and one of them does not contain a $$. The subsequent command will be absorbed into the offending line, almost surely leading to some kind of error message. For example, suppose the illustrative commands we used above were written as follows: Note that the ending $$ is missing from the second command.

\begin{verbatim}
SAMPLE ; 1 – 100 $
CREATE ; x = Rnn(0,1) ; y = x + Rnn(0,1)
REGRESS ; Lhs = y ; Rhs = one,x $
\end{verbatim}

This command sequence produces a string of errors:

```
Error 623: Check for error in ONE,X
Error 623: Look for: Unknown names, pairs of operators, e.g., *
Error 61: Compilation error in CREATE. See previous diagnostic.
```

The problem is that the **REGRESS** command has become part of the **CREATE** command, and the errors arise because this is now not a valid **CREATE** instruction.

### 4.3 Naming Conventions and Reserved Names

Most commands refer to entities such as variables, groups of variables, matrices, procedures, and particular scalars by name. Note in our examples, we have referred to ‘x,’ ‘y,’ abd ‘capital.’ Your data are always referenced by variable names. The requirements for names are:

• They must begin with a letter. Remember that **LIMDEP** is *not* case sensitive. Therefore, you can mix upper and lower case in your names at will, but you cannot create different names with different mixes. E.g., GwEn is the same as GWEn, gwen and GWEN.
- You should not use symbols other than the underscore (‘_’) character and the 26 letters and 10 digits in your names. Other punctuation marks can cause unexpected results if they are not picked up as syntax errors.

- Names may not contain more than eight characters.

There are a few reserved words which you may not use as names for variables, matrices, scalars, namelists, or procedures. These are:

- `one` (used as a variable name, the constant term in a model),
- `b, varb, sigma` (used as matrices, to retain estimation results from all models),
- `n` (always stands for the current sample size),
- `pi` (the number 3.14159...),
- `_obsno` (observation number in the current sample, used by CREATE),
- `_rowno` (row number in data set, used by CREATE),
- `s, sy, ybar, degfrdm, lmda, logl, nreg, rho, rsqrd, ssqrd, sumsqdev` (scalars retained after regressions are estimated),
- `exitcode` (used to tell you if an estimation procedure was successful).

Several of the reserved names are displayed in the project window. Note in Figure 4.1 that there are ‘keys’ next to the three matrix names `b, varb` and `sigma`. These names are ‘locked,’ i.e., reserved. You may not change these entities – for example, you may not create a matrix named `b`. That name is reserved for program use. You are always protected from name conflicts that would arise if you try to name a variable or a matrix with a name which is already being used for something else, such as a matrix or scalar, or if you try to use one of the reserved names. For example, you may not name a variable ‘s’; this is reserved for the standard deviation of the residuals from a regression. LIMDEP will give you a diagnostic if you try to do so, and decline to carry out the command.
4.4 Command Builders

The desktop provides menus and dialog boxes for the functions that you need to manage your data, open projects and text windows, and so on. In almost most all cases, there are commands that you can use for these same functions, though generally you will use the menus. The reverse is true when you analyze your data and do statistical and econometric analysis. You will usually use commands for these procedures. However, we note, for most statistical operations, there are also menus and dialog boxes if you prefer to use them.

For an example, in Figure 4.2, you can see an editing window that contains four commands, a SAMPLE command, followed by two CREATE commands that create a simulated data set on ‘X; and ‘Y.’ and a regression command. The first three commands have already been executed, and the data, as you can see in the project window, are already created. I can now compute the regression by highlighting the REGRESS command and pressing the GO button.

![Figure 4.2 Regression Command](image)

An alternative way to request the regression is to use the Model menu on the desktop and locate the command builder for the linear regression. This is shown in Figure 4.3. When I select this item from the menu, this will open a dialog box that lets me construct my linear regression model without typing the commands in the editor. The command builder is shown in Figure 4.4.
Figure 4.3 Model Menu

Figure 4.4 Regression Command Builder
After selecting the variables that I want to appear on the left and right hand sides of my regression equation, I press Run, and the regression is computed. The results are shown in Figure 4.5.

Figure 4.5 Regression Results in Output Window

The dialog box just shown is called a ‘Command Builder’ because in addition to submitting the necessary information to the program to carry out the regression, it literally creates the command you have issued, and places it in the output window. You can see this result in Figure 14.6, where the ‘REGRESS’ command is echoed in the output window before the results. You can edit/copy and paste this command to move it to your editing window where you may change it and use it as if you had typed it yourself.
Figure 4.6 Command Builder Output
This page intentionally left blank.
Chapter 5
Program Output

5.1 The Output Window

*LIMDEP* will automatically open an output window and use it for the display of results produced by your commands. Figure 5.1 below shows an example. The output window is split into two parts. In the lower part, an echo of the commands and the actual statistical results are accumulated. The upper part of the window displays the TRACE.LIM file as it is being accumulated. Note that there are two tabs in the upper window. You have two options for display in this window. The Trace display is as shown below. If you select the Status tab, instead, this window will display technical information during model estimation, such as the iterations, line search, and function value during maximum likelihood estimation, and execution time if you have selected this option from the Project:Project Settings/Execution tab as well.

![Output Window](image-url)

Figure 5.1 Output Window
5.2 Editing Your Output

The output window provides limited capability for editing. You can select, then delete any of the results in the window. You can also highlight, then use cut or copy in the output window.

But, there is a way to get full editing capability. You can select, then cut or copy the material from the output window and paste it into an editing window (or into any other program, such as a word processor, that you might be using at the same time). The editing window then provides full editing capability, so you can place any annotation in the results that you like. You can save the contents of the editing window as an ordinary text file when you exit *LIMDEP*.

**TIP:** If you wish to extract from your output window a little at a time, one approach is to open a second editing window, and use it for the output you wish to collect. You may have several editing windows open at any time.

5.3 Exporting Your Output.

When you wish to use your statistical results in a report, you will generally copy the results from your output window directly into your word processor. Results in the window are displayed in the default 9 point Courier New font. (You can change this to any font you wish using Tools:Options/Editor.) All numerical results are shown in this form. Graphical results may also be copied directly off the screen. They are produced in a “.wmf” (windows metafile) format. You can paste a graph into your word processor, then resize the figure to whatever size you like. An example appears below in Figure 5.2. An extensive application appears in the next chapter.

![Figure 5.2 Figure Exported From LIMDEP Output](image-url)
6.1 Application – An Econometrics Problem Set

This chapter will illustrate most of the features in LIMDEP that you will use for basic econometric analysis. We will use a simulated problem set to motivate the computations. This application will use many of the commands and computational tools in the program. Of course, we have not documented these yet – they appear later in the manual in the succeeding chapters. We do anticipate, however, that you will be able to proceed from this chapter alone to further use of the program.

6.2 Assignment: The Linear Regression Model

The data listed below are a set of yearly time series observations (1953-2004) on the U.S. Gasoline Market. Use these data to perform the following analyses:

1. Read the raw data into LIMDEP.
2. Obtain descriptive statistics (means, standard deviations, etc.) for the raw data in your data set.
3. The data are in levels. We wish to fit a constant elasticity model, which will require that the variables be in logarithms. Obtain the following variables:
   \[ \log G = \text{log of per capital gasoline consumption} \]
   \[ \log P_g = \text{log of the price (index) of gasoline} \]
   \[ \log I = \text{log of per capita income} \]
   \[ \log P_{nc} = \text{log of price (index) for new cars} \]
   \[ \log P_{uc} = \text{log of price (index) for used cars} \]
   \[ \log P_{pt} = \text{log of price (index) for public transportation} \]
   \[ t = \text{time trend = year – 1952} \]
4. We notice immediately that if we intend to use the (logs of the) price indexes in our regression model, there is at least the potential for a problem of multicollinearity. As a preliminary indication of how serious the problem is likely to be, obtain a time series plot of the four price series.
5. We begin with a simple ‘demand’ equation. Obtain the least squares regression results in a regression of \( \log G \) on the log of the price index, \( \log P_g \). Report your regression results
   a. Notice that the coefficient on log price is positive. Shouldn’t a demand curve slope downward? What is wrong here?
   b. Now, add the obviously missing income variable to the equation. Compute the linear regression of \( \log G \) on \( \log I \) and \( \log P_g \). Report your results and comment on your findings.
The full regression model that you will explore for the rest of this exercise is

\[ \log G = \beta_1 + \beta_2 \log P_g + \beta_3 \log I + \beta_4 \log P_{nc} + \beta_5 \log P_{uc} + \beta_6 \log P_{pt} + \beta_7 t + \epsilon \]

Obtain the least squares estimates of the coefficients of the model. Report your results. Obtain a plot of the residuals as part of your analysis.

a. (For more advanced courses) Compute the least squares regression coefficients, \( \hat{b}, \hat{s}^2 \), the covariance matrix for \( \hat{b} \), and \( R^2 \) using matrix algebra.

b. Test the hypothesis that all of the coefficients in the model save for the constant term are zero using an F test. Obtain the sample F and the appropriate critical value for the test. Use the 95% significance level.

Still using the full model, use an F test to test the hypothesis that the coefficients on the three extra prices are all zero.

Test the hypothesis that the elasticities with respect to the prices of new and used cars are equal.

a. Do the test using an F test.

b. Use a t test, showing all computations.

The market for fossil fuels changed dramatically at the end of 1973. We will examine whether the data contain clear evidence of this phenomenon.


b. (More advanced) Use a Wald test to test the hypothesis. How do the assumptions underlying these two tests differ?

In the analysis to follow, we will work through the problem set in detail. You can work through the steps with the presentation by typing the commands into the text editor. To save time typing, you can just upload the file Tutorial.lim that you will find in the C:\LIMDEP9\LIMDEP Command Files folder. The data set that we will use is also provided as file Gasoline.txt and Gasoline.xls in the folder C:\LIMDEP9\LIMDEP Data Files.

The data for the exercise are listed on the following page.
Application and Tutorial
Year
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004

GasExp
7.4
7.8
8.6
9.4
10.2
10.6
11.3
12.0
12.0
12.6
13.0
13.6
14.8
16.0
17.1
18.6
20.5
21.9
23.2
24.4
28.1
36.1
39.7
43.0
46.9
50.1
66.2
86.7
97.9
94.1
93.1
94.6
97.2
80.1
85.4
88.3
98.6
111.2
108.5
112.4
114.1
116.2
120.2
130.4
134.4
122.4
137.9
175.7
171.6
163.4
191.3
224.5

GasPrice Income
16.668
8883
17.029
8685
17.210
9137
17.729
9436
18.497
9534
18.316
9343
18.576
9738
19.112
9770
18.924
9843
19.043
10226
18.997
10398
18.873
11051
19.587
11430
20.038
11981
20.700
12418
21.005
12932
21.696
13060
21.890
13567
22.050
14008
22.336
14270
24.473
15309
33.059
15074
35.278
15555
36.777
15693
38.907
15991
40.597
16674
54.406
16843
75.509
16711
84.018
17046
79.768
17429
77.160
17659
76.005
18922
76.619
19622
60.175
19944
62.488
19802
63.017
20682
68.837
21048
78.385
21379
77.338
21129
77.040
21505
76.257
21515
76.614
21797
77.826
22100
82.596
22506
82.579
22944
71.874
24079
78.207
24464
100.000
25380
96.289
25449
90.405
26352
105.154
26437
123.901
27113

6-3
PNewCar PUsedCar PPubTrn Pop
47.2
26.7
16.8 159565
46.5
22.7
18.0 162391
44.8
21.5
18.5 165275
46.1
20.7
19.2 168221
48.5
23.2
19.9 171274
50.0
24.0
20.9 174141
52.2
26.8
21.5 177130
51.5
25.0
22.2 180760
51.5
26.0
23.2 183742
51.3
28.4
24.0 186590
51.0
28.7
24.3 189300
50.9
30.0
24.7 191927
49.7
29.8
25.2 194347
48.8
29.0
26.1 196599
49.3
29.9
27.4 198752
50.7
30.7
28.7 200745
51.5
30.9
30.9 202736
53.0
31.2
35.2 205089
55.2
33.0
37.8 207692
54.7
33.1
39.3 209924
54.8
35.2
39.7 211939
57.9
36.7
40.6 213898
62.9
43.8
43.5 215981
66.9
50.3
47.8 218086
70.4
54.7
50.0 220289
75.8
55.8
51.5 222629
81.8
60.2
54.9 225106
88.4
62.3
69.0 227726
93.7
76.9
85.6 230008
97.4
88.8
94.9 232218
99.9
98.7
99.5 234333
102.8
112.5
105.7 236394
106.1
113.7
110.5 238506
110.6
108.8
117.0 240683
114.6
113.1
121.1 242843
116.9
118.0
123.3 245061
119.2
120.4
129.5 247387
121.0
117.6
142.6 250181
125.3
118.1
148.9 253530
128.4
123.2
151.4 256922
131.5
133.9
167.0 260282
136.0
141.7
172.0 263455
139.0
156.5
175.9 266588
141.4
157.0
181.9 269714
141.7
151.1
186.7 272958
140.7
150.6
190.3 276154
139.6
152.0
197.7 279328
139.6
155.8
209.6 282429
138.9
158.7
210.6 285366
137.3
152.0
207.4 288217
134.7
142.9
209.3 291073
133.9
133.3
209.1 293951


6.3 Read the Raw Data

In order to analyze your data, you must “read” them into the program. There are many ways to read your data into LIMDEP. We will consider two simple ones, via an Excel spreadsheet and as portable ASCII text. Others are described in the full manual for the software.

Excel Spreadsheet

If your data are given to you in the form of an Excel spreadsheet, then you can “import” them directly into the program. The first step is to use the Project menu on the desktop as shown in Figure 6.1. This would then launch a mini-explorer. You would then locate your file in the menu. Finally, you can just double click the file name and the data are read into the program.

![Figure 6.1 Reading an Excel Spreadsheet File](image)

**WARNING**: For at least 15 years, until 2007, the .XLS format has served as a lingua franca for exchanging data files between programs. Almost any statistical package could read one. Beginning in 2007, Microsoft drastically changed the internal format of spreadsheet files, and Excel 2007 files are no longer compatible with other software. It will be quite a while (if ever) for software authors to catch up with this change, so for the present, we note that LIMDEP cannot read an Excel 2007 spreadsheet file. If you are using Excel 2007, be sure to save your spreadsheets in the 2003 format.
Text Input File

A second, equally simple way to read your data is simply to read them off the screen in a text editor. The file `GasolineData.lim` contains text that is exactly the highlighted command, names and data that appear in Figure 6.3. To get these data into the program, we can proceed as follows: First, we start LIMDEP. Then, use File: Open and navigate to the file. You will reach the file in the mini-explorer.

![Figure 6.2 Mini-explorer for Data File](image)

Now, just double click the file name, and the contents of the file will be placed in a new text editing window as shown in Figure 6.3. With the data in this form, all that is needed now is Edit/Select All, then click the GO button, in the desktop toolbar, and the data will be read from the screen.
For purposes of this method, suppose you have been given the data table in a document file that will allow you to copy and paste the data file exactly the way they appear on the page. Figure 6.4 below shows the example for our tutorial. The data that we wish to analyze appear on the left hand page. To transport these data into LIMDEP, we could proceed as follows: First, start LIMDEP. Then, we open a text editor in LIMDEP with File:New/Text/Command Document Type the command “READ$” in the top line. (Do not forget the $ at the end of the command.) Finally, just highlight the data (and names line) in the document, use Edit/Copy in the word processor and Edit/Paste in LIMDEP.

The end result of either of these approaches will be to place the data in the text editor, ready for you to use.

Running the Data File as a Command Set

Placing the READS command and data in the text editor, then highlighting the material and clicking GO to execute the READS actually involves an extra step. You can instruct LIMDEP to do the whole thing at once, since the file GasolineData.lim contains the command and the data. If you use Run:Run File from the desktop menu, then select the .lim file from the mini-explorer, LIMDEP will read the file and internally, by itself, highlight the material and press its own GO button. This process will carry out the READS command and proceed directly to the confirmation shown in Figure 6.5.
The confirmation in the output window and the appearance of the names of the variables in the project window indicate that the raw data have been read and we can now analyze them.
You can verify that the data have been properly imported by looking at the data editor. This is a small spreadsheet editor. You might use it to enter a small data set with a few observations by typing them directly into the program’s memory. The data editor will display the data that have already been placed in memory. To open the data editor, click the Data Editor button on the desktop toolbar, as shown in Figure 6.6.

**Figure 6.6 Data Editor**

**Rereading the Data**

Though each one of these methods of reading the data into the program is short and simple, it might still seem like a bit of effort just to get the data ready to use. The first time you use a data set, there has to be some way to get the data ready for this program to use them. You could just type them in yourself – there is a spreadsheet style editor as shown in Figure 6.6. But, if the data already exist somewhere else, that would be terribly inefficient and inconvenient. We do emphasize, however, that however you enter your data set into LIMDEP the first time, you will only do it once. Once the data are in the program and in a project, you will save that project in a file. Thereafter, when you want to use these data again, you will simply double click the project, in a mini-explorer, on your desktop, in LIMDEP’s recently used files, or somewhere else, and you will be ready to go back to work. See Section 6.5.3 for further discussion.
6.4 Tutorial Commands

In this tutorial, you will use nearly all of the features of LIMDEP that you will need to complete a graduate course in econometrics. We will run through a long sequence of operations one at a time, and display the results at each step. You will want to carry out these operations as we do. Rather than type them one at a time, it will be more convenient for you to have them available in a text editor, where you can execute each one just by highlighting one or more lines and clicking the GO button.

The listing below contains all of the commands that we used in this tutorial. You can place them in your screen editor by using File:Open then making your way to

C:\LIMDEP9\LIMDEP Command Files\Tutorial.lim

Select this file. Your screen will then contain all the commands that will be used in the tutorial to follow. (The text editor does not replicate the boldface in the listing below.) You can then ‘follow along’ by highlighting and executing each command or set of commands as they are discussed below.

Figure 6.6 Tutorial Commands in Text Editor
These are the commands contained in the tutorial command file. Lines that begin with a question mark are comments. If you submit these lines to the program as if they were commands, they are ignored.

? Read the raw data

\[
\text{READ $}
\]

<table>
<thead>
<tr>
<th>Year</th>
<th>GasExp</th>
<th>GasPrice</th>
<th>Income</th>
<th>PNewCar</th>
<th>PUsedCar</th>
<th>PPubTrn</th>
<th>Pop</th>
</tr>
</thead>
<tbody>
<tr>
<td>1953</td>
<td>7.4</td>
<td>16.668</td>
<td>8883</td>
<td>47.2</td>
<td>26.7</td>
<td>16.8</td>
<td>159565</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2004</td>
<td>224.5</td>
<td>123.901</td>
<td>27113</td>
<td>133.9</td>
<td>133.3</td>
<td>209.1</td>
<td>293951</td>
</tr>
</tbody>
</table>

? Descriptive Statistics for all variables

\[
\text{DSTAT ; Rhs = * $}
\]

? Transformed Variables

\[
\text{CREATE ; logg = log (gasexp/(pop*gasprice))}
\]

\[
\text{; logpg = log(gasprice)}
\]

\[
\text{; logi = log(income)}
\]

\[
\text{; logpnc = log(pnewcar)}
\]

\[
\text{; logpuc = log(pusedcar)}
\]

\[
\text{; logppt = log(ppubtrn)}
\]

\[
; t = (year \, - \, 1953) $
\]

? Define data to be yearly time series and set first year as 1953, observations 1953 to 2004.

\[
\text{DATES ; 1953 $} \quad \text{(This indicates the first year of the data)}
\]

\[
\text{PERIOD ; 1953 \, – \, 2004 $} \quad \text{(This indicates which years of data to use)}
\]

? Time series of four price variables with title and grid for readability

\[
\text{PLOT ; Rhs = gasprice, pnewcar, pusedcar, ppubtrn}
\]

\[
; \text{Grid}
\]

\[
; \text{Title=Time Series Plot of Price Indices $}
\]

? Three different ways to display a correlation matrix for four variables

\[
\text{DSTAT ; Rhs = gasprice, pnewcar, pusedcar, ppubtrn ; Output = 2 $}
\]

\[
\text{MATRIX ; list ; Xcor (gasprice, pnewcar, pusedcar, ppubtrn) $}
\]

\[
\text{NAMELIST ; prices = gasprice, pnewcar, pusedcar, ppubtrn $}
\]

\[
\text{MATRIX ; list ; Xcor (prices) $}
\]

? Simple regression of logG on a constant and log price. Computations to analyse potential biases based on the left out variable formula and plausible values.

\[
\text{REGRESS ; Lhs = logg ; Rhs = one, logpg $}
\]

\[
\text{CALC ; list ; spi = Cov(logPg, logI)}
\]

\[
\text{; spp = Var(logPg)}
\]

\[
; plim = -0.1 + spi/spp \times 1.0 $\]

? Multiple regression including both price and income to confirm expectations about what happens when income is omitted from the equation. Includes a plot of residuals.

\[
\text{REGRESS ; Lhs = logG ; Rhs = one, logpg, logi ; Plot residuals $}
\]

? Full regression model including all variables

\[
\text{NAMELIST ; x1 = logpg, logl}$
\]

\[
; x2 = logpnc, logpuc, logppt$
\]

\[
; x = one, x1, x2, t $\]

\[
\text{REGRESS ; Lhs = logg ; Rhs = x ; Plot residuals $}
\]

? Least squares computations using matrix algebra, slopes, standard error, R squared, covariance matrix for coefficients. Results are displayed in a table.

\[
\text{MATRIX ; bols = \langle X'X \rangle \times X \logg $}
\]

\[
\text{MATRIX ; e = logg - X \times bols $}
\]

\[
\text{CALC ; list ; se = sqrt((e' e) / (n-col(X))) ; r2 = 1 - e'e/(n-1) \times var(logg) $}
\]

\[
\text{MATRIX ; v = s2 \times \langle X'X \rangle \; \; \; \text{Stat (bols, v, x) $}
\]
? F statistic for the hypothesis that all coefficients are zero. R squared for the regression
    CALC ; list ; f = (r2/(col(X)-1)) / ((1-r2)/(n-col(X))) $
    CALC ; list ; se = sqr(sumsqdev / degfrdm)
    ; r2 = 1 - e'e/((n-1)*var(logg)) $

? F statistic using results retained by the regression. Critical values for F and t.
    CALC ; list ; f = rsqrd/(k-1)/((1-rsqrd)/degfrdm) $
    CALC ; list ; fc = Ftb(.95,(kreg-1),degfrdm)
    ; tc = ttb(.95, (n-col(x))) $

? Constrained least squares, three coefficients constrained to equal zero. Test of
? hypothesis using F statistic. Two ways: Built in and using the R squareds.
    REGRESS ; Lhs = logg ; Rhs = x ; Cls: b(4) = 0, b(5) = 0, b(6) = 0 $  
    NAMELIST ; xu = x ; xr = one,x1,t $  
    REGRESS ; Lhs = logg ; Rhs = xu $  
    CALC ; rsqu = rsqrd $  
    REGRESS ; Lhs = logg ; Rhs = xr $  
    CALC ; rsqr = rsqrd
    ; list ; f = ((rsqu – rsqr)/Col(x2))/((1-rsqu)/(n-Col(x))) $

? Use the built in calculator function to compute R squared
    CALC ; rsqu = rsq(Xu,logg)  
    ; rsqr = rsq(Xr,logg)  
    ; list ; f = ((rsqu – rsqr)/Col(x2))/((1-rsqu)/(n-Col(x))) $

? Testing the hypothesis using the Wald statistic. Built in command then using matrix algebra.
    REGRESS ; Lhs = logg ; Rhs = x $  
    WALD ; Fn1 = b_logpnc-0
    ; Fn2 = b_logpuc-0
    ; Fn3 = b_logppt-0 $  

? Compute the regression. Then use the saved matrices.
    REGRESS ; Lhs = logg ; Rhs = x $  
    MATRIX ; b2 = b(4:6)  
    ; v22 = Varb(4:6,4:6)  
    ; list ; W = b2'<v22>b2 $  
    REGRESS ; Lhs = logg ; Rhs = x ; Cls: b(4) – b(5) = 0 $  

? Compute a restricted least squares estimator using matrix algebra.
    MATRIX ; r = [0,0,0,1,0,0,0 / 0,0,0,0,1,0,0 / 0,0,0,0,0,1,0] ; q = [0/0/0] $  
    MATRIX ; bu = <X'X>*X'y ; C = R'*<X'X>*R' ; d = R*b - q
    ; bc = bu - <X'X>*R'*<C>*d $  

? Test the hypothesis that two coefficients are equal using a t test.
? This could be built directly into the REGRESS command with ;Cls:b(b(-b(5)=0.
    REGRESS ; Lhs = logg ; Rhs = x $  
    CALC ; list ; tstat = (b(4)-b(5)) / sqr(varb(4,4)+varb(5,5)-2*varb(4,5))  
    ; tc = ttb (.975, degfrdm)
    ; pvalue = 2*(1-tds(tstat,degfrdm)) $

? Chow test of structural change. For each subperiod and for the full period, obtain the residual
? sum of squares (without displaying the whole regression).
    PERIOD ; 1953-2004 $  
    CALC ; ssep = Ess(x,logg) ; k = col(x) $  
    PERIOD ; 1953 – 1973 $  
    CALC ; sse1 = Ess(x,logg) ; n1 = n $
PERIOD ; 1974 – 2004 $
$CALC ; sse2 = Ess(x,logg) ; n2 = n $
$CALC ; list ; f = ((ssep - (sse1+sse2))/k) / ((sse1+sse2)/(n1+n2-2*k)) $
$Ftb(.95,k,(n1+n2-2*2))$

Structural change test using Wald approach requires separate regressions and some matrix algebra.

PERIOD ; 1953 – 1973 $

Compute regressions quietly as I am not interested in seeing the results.

REGRESS ; Lhs = logg ; Rhs = x ; quietly $

MATRIX ; b1 = b ; v1 = varb $

PERIOD ; 1974 – 2004 $

REGRESS ; Lhs = logg ; Rhs = x ; quietly $

MATRIX ; b2 = b ; v2 = varb $

Compute Wald statistic. Then display critical value from chi squared table.

MATRIX ; db = b1 – b2 ; vdb = v1 + v2 $
$; list ; w = db’<vdb>db $

CALC ; list ; cstar = ctb(.95,k) $

6.5 Using LIMDEP for Linear Regression Analysis

The following analyses show how to use LIMDEP for the standard applications in analysis of the linear model.

6.5.1 Obtain Descriptive Statistics

The command for obtaining descriptive statistics is

DSTAT ; Rhs = the variables to be described $

There is a useful short hand; the ‘wildcard’ character ‘*’ in this context means ‘all variables.’ So, we will use the simpler instruction,

DSTAT ; Rhs = * $

The results are shown in the output window shown in Figure 6.7.
6.5.2 Transformed Variables

The command for computing transformed variables is CREATE. A single command can be used for all of the transformations listed,

```
CREATE ; logg  =log (gasexp/(pop*gasprice))
; logpg = log(gasprice)
; logi = log(income)
; logpnc = log(pnewcar)
; logpuc = log(pusedcar)
; logppt = log(ppubtrn)
; t = Year – 1953 $
```

After we place this command in the text editor and execute it, the project expands to include the new variables. The new project window in shown in Figure 6.8.
6.5.3 Saving and Retrieving Your Project

Since you have read your data into the program and created some new variables, this is a good time to save your project. Select File:Save Project As… The mini-explorer that appears next allows you to save the file anywhere you like. The Project Files folder is a natural choice. See Figures 6.9 and 6.10 for our application.

![Figure 6.9 Saving the Current Project](image)

![Figure 6.10 Mini-explorer for Saving the Project](image)
When you restart LIMDEP later, you can use the File menu to retrieve your project file and resume your work where you left off before. The File menu is shown in Figure 6.11. Note that once we have saved our project file, the file name will reappear in the Recently Used part of the File menu.

![Figure 6.11 Reloading the Project](image)

As with any software, it is a good idea to save your work this way periodically.
6.5.4 Time Series Plot of the Price Variables

The **PLOT** instruction is used to produce the different types of graphs you will obtain. Since these are time series data, we will first inform **LIMDEP** of that fact. The commands are

- **DATES** ; 1953 $ (This indicates the first year of the data)
- **PERIOD** ; 1953 – 2004 $ (This indicates which years of data to use)

Then, we generate our plot with the command

- **PLOT** ; Rhs = gasprice, pnewcar, pusedCar, ppubtrn
  ; Grid
  ; Title=Time Series Plot of Price Indices $

It is clear from Figure 6.12 that although the price series are correlated – that is in the nature of aggregate long period time series data – the correlations between the gasoline price and the public transport price seem unlikely to be severe enough to seriously impact the regression. However, the new and used car price indexes are quite highly correlated, and one might surmise that together in a regression, it would be difficult to resolve separate impacts of these two variables.

![Figure 6.12 Time Series Plots](image)

To pursue the issue a bit, we will compute a correlation matrix for the four price variables. As usual, there is more than one way to proceed. Here are two: First, the **DSTAT** command can be extended to request correlations by simply adding ; **Output = 2** to the instruction. Thus, we might use
This produces the results in Figure 6.13; the descriptive statistics now include the correlations among the price variables.

![Figure 6.13 Descriptive Statistics with Correlations](image)

It is possible to obtain these results with the dialog boxes in the command builder. We begin with Model:Data Description:Descriptive Statistics on the desktop menu shown in Figure 6.14 and 6.15.

![Figure 6.14 Model Menu for Descriptive Statistics](image)
The variables are specified on the Main page, then the correlation matrix is requested as an option on the Options page. Clicking Run then produces the results that appear in Figure 6.13.

A second approach is to compute and display the correlation matrix. To do this, since we are computing a correlation matrix, we use the MATRIX command. There is a matrix function that computes correlations, which would appear as follows:

```
MATRIX ; list ; Xcor (gasprice,pnewcar,pusedcar,ppubtrn) $
```

This produces the following results in the output window.

```
Correlation Matrix for Listed Variables
GASPRICE PNEWCAR PUSEDCAR PPUBTRN
GASPRICE 1.0000 .93605 .92277 .92701
PNEWCAR .93605 1.0000 .99387 .98074
PUSEDCAR .92277 .99387 1.0000 .98242
PPUBTRN .92701 .98074 .98242 1.0000
```

which are, of course the same as those with the descriptive statistics.

There is another convenient feature of MATRIX that we should note at this point. The NAMELIST command below associates the name ‘prices’ with the four price variables:

```
NAMELIST ; prices = gasprice,pnewcar,pusedcar,ppubtrn $
```

Now, in any setting where we wish to use these four variables, we can use the name prices instead. This defines a data matrix with these four columns. Thus, we can now shorten the MATRIX command to

```
MATRIX ; list ; Xcor (prices) $
```

Finally, note that each MATRIX command begins with ;list. This requests that the result of the matrix computation be displayed on the screen in the output window. Why is this needed? MATRIX is part of the programming language. You might be writing sets of commands that do matrix computations that are intermediate results to be used later that you are not necessarily interested in seeing. If you do not request LIMDEP to list the results, the program assumes this is the case. Several examples below will illustrate this.
6.5.5 Simple Regression

The REGRESS command is used to compute a linear regression. (As usual, you can also use the Model menu and command builders, however, we will generally use the commands.) The essential instruction is

\[
\text{REGRESS ; Lhs = one,logpg}
\]

The regression results are displayed in the output window shown in Figure 6.16.

**NOTE**: The right hand side of the REGRESS command contains two ‘variables,’ one and the variable that we wish to appear in the model. The ‘one’ is the constant term in the model. If you wish for your model to include a constant term – and this should be in the vast majority of cases – you request it by including one on the right hand side. LIMDEP does not automatically include a constant term in the model. In fact, most programs work this way, though there are a few that automatically put a constant term in the model. Then, it becomes inconvenient to fit a model without one. In general, LIMDEP requires you to specify the model the way you want it; it does not make assumptions for you.
The regression output shows LIMDEP’s standard format. The set of diagnostics, including $R^2$, $s^2$, sum of squares, F statistic for the regression, and so on are shown above the regression coefficients. The coefficients are reported with standard errors, $t$ ratios, ‘$p$ values’ and the means of the associated independent variables.

For the specific application, we note, the estimated price elasticity is, indeed, positive, which is unexpected for economic data. Two explanations seem natural. First, assuming we are estimating a demand equation, then a demand equation should include income as well as price. In this particular market, in fact, it is well known that demand is strikingly inelastic with respect to price, and that income is the primary driver of consumption. The estimated equation is missing a crucial variable. The theoretical result for a demand equation from which income has been omitted would be as follows:

Model:  

$$
\log G = \alpha + \beta_{\text{Price}} \log Price + \beta_{\text{Income}} \log Income + \varepsilon
$$

$$
\text{plim } b_{\text{Price}} = \beta_{\text{Price}} + \frac{\text{Cov}(\log Price, \log Income)}{\text{Var}(\log Price)} \beta_{\text{Income}}
$$

where $b_{\text{Price}}$ is the simple least squares slope in the regression of $\log G$ on a constant and $\log Pg$. It is clear from the data that the two terms in the fraction are positive. If the income elasticity were positive as well, then the result shows that the estimator in this short regression is biased upward toward and possibly even past zero. We could construct some evidence on what might be expected here. Suppose the income elasticity were +1.0 and the price elasticity were -0.1. The terms in the fraction can be obtained with the command

```
CALC ; list ; spi = Cov(logpg,logi) ; spp = Var(logpg) ; plim = -0.1 + spi/spp * 1.0 $
```

The result is shown here. If the theory is right, then even if the price elasticity really is negative, with these time series data and the positive income elasticity, we should observe a positive coefficient on $\log Pg$ in the simple regression (which we did).

```
+------------------------------------+
| Listed Calculator Results          |
+------------------------------------+
SPI     =       .223618
SPP     =       .461029
PLIM    =       .385040
Calculator: Computed   3 scalar results
```

We proceed to compute the more appropriate regression which contains both price and income (and which will either confirm or refute our theory above). We do note the second explanation for the finding, however. We have presumed so far that we are actually fitting a demand curve, and that we can treat the price as exogenous. If the price were determined on a world market, this might be reasonable, but that is less so in the U.S. which does manipulate the price of gasoline. We leave for others to resolve this issue, and continue with the multiple regression.

The more complete model is estimated with the command

```
REGRESS ; Lhs = logG ; Rhs = one,logpg,logi $
```
The results shown below are more in line with expectations, and conform fairly closely to the hypothetical exercise done earlier. The data from this period do suggest that the income elasticity is close to one and the price elasticity is indeed negative and around -0.17. The fit of the model is extremely good as well.

+----------------------------------------------------+
| Ordinary least squares regression                   |
| LHS=LOGG  Mean = -12.24504                           |
|         Standard deviation = .2388115                |
| WTS=none  Number of observs. = 52                    |
| Model size Parameters = 3                           |
|         Degrees of freedom = 49                      |
| Residuals Sum of squares = .1849006                 |
|         Standard error of e  = .6142867E-01          |
| Fit      R-squared = .9364292                        |
|         Adjusted R-squared = .9338345                |
| Model test F[  2,    49] (prob) = 360.90 (.0000)    |
| Diagnostic Log likelihood = 72.83389                 |
|         Restricted(b=0) = 1.188266                   |
|         Chi-sq [  2] (prob) = 143.29 (.0000)         |
| Autocorrel Durbin-Watson Stat. = .1168578           |
|         Rho = cor[e,e(-1)] = .9415711               |
+----------------------------------------------------+

| Variable| Coefficient | Standard Error |t-ratio |P[|T|>t]| Mean of X|
|---------|-------------|----------------|--------|--------|----------|
| Constant| -20.9557732 | .59398134      -35.280  .0000          |
| LOGPG   | -.16948546  | .03865426      -4.385   .0001  3.72930296|
| LOGI    | .96594886   | .07529145      12.829   .0000  9.67214751|

As a final check on the model, we have plotted the residuals by adding ;PlotResiduals to the REGRESS command.

REGRESS ; Lhs = logG ; Rhs = one,logpg,logi ; Plot residuals $  

The results are striking:.  The residuals are far from a random sequence.  The long sequences of negative then positive then negative residuals suggests that something is clearly missing from the equation.

![Figure 6.17 Residual Plot](image)
6.5.6 Multiple Regression

For the full model, we use the following setup, which combines some of our earlier commands.

\begin{verbatim}
NAMELIST ; x1 = logpg,logI
; x2 = logpnc,logpuc,logppt
; x = one, x1,x2,t $
\end{verbatim}

Notice the third namelist is constructed by combining the first two and adding a variable to the list. The regression is then computed using

\begin{verbatim}
REGRESS ; Lhs = logg ; Rhs = x ; Plot residuals $
\end{verbatim}

The full results are as follows:

+----------------------------------------------------+
| Ordinary least squares regression                  |
| LHS=LOGG Mean                                      =  -12.24504 |
| Standard deviation                                 =   .2388115 |
| WTS=none Number of observs.                       =         52 |
| Model size Parameters                              =          7 |
| Degrees of freedom                                 =         45 |
| Residuals Sum of squares                           =   .1014368 |
| Standard error of e                                =   .4747790E-01 |
| Fit R-squared                                      =   .9651249 |
| Adjusted R-squared                                 =   .9604749 |
| Model test F[  6,    45] (prob)                    = 207.55 (.0000) |
| Diagnostic Log likelihood                          =   88.44384 |
| Restricted(b=0)                                    =   1.188266 |
| Chi-sq [  6] (prob)                                = 174.51 (.0000) |
| Autocorrel Durbin-Watson Stat.                     =   .4470769 |
| Rho = corr[e,e(-1)]                               =   .7764615 |
+----------------------------------------------------+

+--------+--------------+----------------+--------+--------+----------+
|Variable| Coefficient  | Standard Error |t-ratio |P[|T|>t]| Mean of X|
+--------+--------------+----------------+--------+--------+----------+
|Constant|  -26.9680492 |  2.09550408    | -12.869 |  .0000 |
|LOGPG   |  -0.05373342 |  0.04251099    |  -1.264 |  .2127 |  3.72930296 |
|LOGI    |   1.64909204 |  0.20265477    |   8.137 |  .0000 |  9.67214751 |
|LOGPNC  |  -0.03199098 |  0.20574296    |  -0.155 |  .8771 |  4.38036654 |
|LOGPUC  |  -0.07393002 |  0.10548982    |  -0.701 |  .4870 |  4.10544881 |
|LOGPPT  |  -0.06153395 |  0.12343734    |  -0.499 |  .6206 |  4.14194132 |
|T       |   -0.01287615|  0.00525340    |  -2.451 |  .0182 |  25.5000000 |
+--------+--------------+----------------+--------+--------+----------+

The coefficient on \textit{logPp} has now fallen to only -0.05. The implication is that after accounting for the uses of gasoline and the price of a major competitor (public transport) and income, the price elasticity of demand for gasoline really does seem to be close to zero. The residuals are still far from random, but are somewhat more in that direction than the earlier ones. This suggests, as surmised earlier, that the specification of the model is in need of improvement.
The least squares coefficients, standard errors, s and $R^2$ are all reported with the standard results. It is also convenient to obtain these results using least squares algebra, as shown below.

\[
\text{MATRIX \ ; \ bols = } <X'X> \ast X \text{ \ logg } \$
\text{MATRIX \ ; \ e = logg - X \ast bols }$
\text{CALC \ ; \ list \ ; \ se = } \sqrt{(e'e) / (n-col(X))}$
\text{; \ r2 = } 1 - e'e/(n-1)*\text{var(logg)}$
\text{MATRIX \ ; \ v = s2 * } <X'X> \$
\text{; \ Stat (bols,v,x) }$

(Note that in this set of commands, we are avoiding using the reserved names $s$ and $r$.)

To test the hypothesis that all the coefficients save for the constant term are zero, we need

\[
F[K-1,n-K] = \frac{R^2/K}{[(1-R^2)/(n-K)]}
\]
The value is reported in the regression results above (207.55). But, it is easy enough to compute it directly, using

\[
\text{CALC} ; \text{list } f = (r2/(col(X)-1)) / ((1-r2)/(n-col(X))) \]

In fact, this can be made a bit easier, because the ingredients we need were automatically computed and retained by the regression command. After the \texttt{REGRESS} command is executed, the new project window is

The new scalars that appear in the project window are:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{ssqrd}</td>
<td>( s^2 )</td>
</tr>
<tr>
<td>\texttt{rsqrd}</td>
<td>( R^2 )</td>
</tr>
<tr>
<td>\texttt{s}</td>
<td>( s )</td>
</tr>
<tr>
<td>\texttt{sumsqdev}</td>
<td>( e'e )</td>
</tr>
<tr>
<td>\texttt{degfrdm}</td>
<td>( n - K )</td>
</tr>
<tr>
<td>\texttt{kreg}</td>
<td>( K )</td>
</tr>
<tr>
<td>\texttt{nreg}</td>
<td>( n )</td>
</tr>
</tbody>
</table>

for the most recent regression computed. Thus, after the regression command, we could have used

\[
\text{CALC} ; \text{list } \text{se} = \text{sqr(sumsqdev/degfrdm)} ; \text{r2} = 1 - e'e/((n-1)*\text{var}('logg')) \]

Of course, \texttt{se} and \texttt{r2} were superfluous, since \texttt{s} and \texttt{rsqrd} are the same values. For computing the F statistic, we could have used

\[
\text{CALC} ; \text{5} = \text{rsqrd}/(k-1)/((1-rsqrd)/\text{degfrdm}) \]

Finally, in order to obtain the critical value for the F test, we use the internal table. For the F statistic and t statistics, we use

\[
\text{CALC} ; \text{list } \text{fc} = \text{Ftb(.95,(kreg-1),degfrdm)} ; \text{tc} = \text{ttb(.95, (n-col(x)))} \]

which produces

<table>
<thead>
<tr>
<th>Listed Calculator Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>FC</td>
</tr>
<tr>
<td>TC</td>
</tr>
</tbody>
</table>
6.5.7 Hypothesis Tests

There are usually several ways to carry out a computation. Here we will test a hypothesis using several tools. The model is

\[ \log G = \beta_1 + \beta_2 \log P_g + \beta_3 \log I + \beta_4 \log P_{nc} + \beta_5 \log P_{uc} + \beta_6 \log P_{pt} + \beta_7 t + \varepsilon \]

The first hypothesis to be tested is that the three cross price elasticities are all zero. That is,

\[ H_0: \beta_4 = \beta_5 = \beta_6 = 0. \]

A direct approach is to build the linear hypothesis directly into the model command. There is a particular specification for this, as shown in the command below:

NAMELIST ; x1 = \logpg,\logi \ ; x2 = \logpnc,\logpuc,\logppt \ ; x =\text{one}, x1,x2,t $
REGRESS  ; \text{Lhs} = \logg \ ; \text{Rhs} = x$
\hspace{1cm} ; \text{Cls: b}(4) = 0, b(5) = 0, b(6) = 0$

This produces the following results: The results of the unrestricted regression are shown first, followed by

<table>
<thead>
<tr>
<th>Linearly restricted regression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ordinary least squares regression</td>
</tr>
<tr>
<td>LHS=LOGG Mean = -12.24504</td>
</tr>
<tr>
<td>Standard deviation = .2388115</td>
</tr>
<tr>
<td>WTS=none Number of observs. = 52</td>
</tr>
<tr>
<td>Model size Parameters = 4</td>
</tr>
<tr>
<td>Degrees of freedom = 48</td>
</tr>
<tr>
<td>Residuals Sum of squares = .1193176</td>
</tr>
<tr>
<td>Standard error of e = .4985762E-01</td>
</tr>
<tr>
<td>Fit R-squared = .9589773</td>
</tr>
<tr>
<td>Adjusted R-squared = .9564134</td>
</tr>
<tr>
<td>Model test F[ 3, 48] (prob) = 374.03 (.0000)</td>
</tr>
<tr>
<td>Diagnostic Log likelihood = 84.22267</td>
</tr>
<tr>
<td>Restricted(b=0) = 1.188266</td>
</tr>
<tr>
<td>Chi-sq [ 3] (prob) = 166.07 (.0000)</td>
</tr>
<tr>
<td>Autocorrel Durbin-Watson Stat. = .4220089</td>
</tr>
<tr>
<td>Rho = cor[e,e(-1)] = .7889955</td>
</tr>
<tr>
<td>Restrictns. F[ 3, 45] (prob) = 2.64 (.0606)</td>
</tr>
<tr>
<td>Not using OLS or no constant. Rsqd &amp; F may be &lt; 0.</td>
</tr>
<tr>
<td>Note, with restrictions imposed, Rsqd may be &lt; 0.</td>
</tr>
</tbody>
</table>

| Variable | Coefficient | Standard Error | t-ratio | P[|T|>t]| Mean of X |
|---------|-------------|----------------|--------|--------|----------|
| Constant| -28.0817710 | 1.46871226 | -19.120 | .0000 |
| LOGPG   | -.13016036  | .03229378 | -4.031 | .0002 | 3.72930296 |
| LOGI    | 1.73922391  | .16247609 | 10.704 | .0000 | 9.67214751 |
| LOGPNC  | .249800D-15 | ...... | ...... | ...... |
| LOGPUC  | -.555112D-16| ...... | ...... | ...... |
| LOGPPT  | -.117961D-15| ...... | ...... | ...... |
| T       | -.01960344  | .00381652 | -5.136 | .0000 | 25.5000000 |

(The coefficients on the three prices in the restricted regression are not exactly zero – the values are pure rounding error.) The \( F \) statistic is on the boundary, not quite statistically significant.
Although the restrictions can be built into the **REGRESS** command, there will be cases in which one is interested in applying the theoretical results directly. The template formula for computing the $F$ statistic is

$$F = \frac{(R^2_{\text{Unrestricted}} - R^2_{\text{Restricted}})}{(1 - R^2_{\text{Unrestricted}})/(n - K)} \div J$$

Where $J$ is the number of restrictions and $K$ is the number of parameters in the full regression with no restrictions. In order to compute the statistic this way, we need the $R^2$'s from the two regressions. For this application, the restricted regression omits the three additional price variables. A way we can proceed is

```
NAMELIST ; x1 = logpg,logI ; x2 = logpnc,logpuc,logppt ; x =one, x1,x2,t
; xu = x  ;  xr = one,x1,t $
REGRESS  ; Lhs = logg ; Rhs = xu $
CALC   ; rsqu = rsqrd $
REGRESS  ; Lhs = logg ; Rhs = xr $
CALC   ; rsqr = rsqrd
; List  ; f = ((rsqu – rsqr)/Col(x2))/((1-rsqu)/(n-Col(x))) $
```

Finally, since we aren’t actually interested in seeing the regressions at this point, there is a more direct way to proceed. The calculator function Rsq(matrix,variable) computes the $R^2$ in the regression of the variable on the variables in the matrix, which is exactly what we need. Thus,

```
CALC   ; rsqu = Rsq(xu,logg) ; rsqr = Rsq(xr,logg)
; List  ; f = ((rsqu – rsqr)/Col(x2))/((1-rsqu)/(n-Col(x))) $
```

An alternative way to compute the test statistic is to use the large sample version, which is a chi squared statistic, the Wald statistic. Formally, the Wald statistic is

$$W = b_2' (W_{22})^{-1} b_2$$

Where $b_2$ is the subset of the least squares coefficient vector, the three price coefficients, and $W_{22}$ is the $3 \times 3$ submatrix of the covariance matrix of the coefficient estimator. This is estimated with $s^2(X'X)^{-1}$. For the linear regression model, the $W$ statistic turns out to be just $J$ times the $F$ statistic, where $J$ is the number of restrictions (three). We can compute this by using the built in program written just for this purpose, as follows:

```
REGRESS  ; Lhs = logg ; Rhs = x $  
WALD     ; Fn1 = b_logpnc-0
; Fn2 = b_logpuc-0
; Fn3 = b_logppt-0 $
```

The **WALD** procedure is used for two purposes, first, to compute estimates of standard errors for functions of estimated parameters and, second, to compute Wald statistics. The result for our model is
+-----------------------------------------------+
| WALD procedure. Estimates and standard errors |
| for nonlinear functions and joint test of     |
| nonlinear restrictions.                       |
| Wald Statistic             =      7.93237     |
| Prob. from Chi-squared[ 3] =       .04743     |
+-----------------------------------------------+

|Variable| Coefficient  | Standard Error |b/St.Er.|P[|Z|>z]| 
|--------|--------------|----------------+--------+--------|
|Fncn(1) |   -.03199098       .20574296     -.155   .8764 |
|Fncn(2) |   -.07393002       .10548982     -.701   .4834 |
|Fncn(3) |   -.06153395       .12343734     -.499   .6181 |

The chi squared statistic is given in the box of results above the estimates of the functions. The statistic given is the statistic for the joint test of the hypothesis that the three functions equal zero.

To illustrate the computations, we consider how to obtain the Wald statistic ‘the hard way,’ that is, by using matrix algebra. This would be

```
REGRESS ; Lhs = logg ; Rhs = x  $

MATRIX  ; b2 = b(4:6)  ; v22 = Varb(4:6,4:6)
; list  ; w = b2'<v22>b2  $
```

Matrix W has  1 rows and  1 columns.

```
1
+--------------
1| 7.93237
```

It is also instructive to see how to compute the restricted least squares estimator using the textbook formulas. The result for computing the least squares estimator \( b^* \) subject to the restrictions \( Rb - q = 0 \) is

\[
b^* = b - (XX')^{-1}R'[R(XX')^{-1}R']^{-1}(Rb - q)
\]

where \( b \) is the unconstrained estimator. To compute this using matrix algebra, we use

```
MATRIX  ; r = [0,0,0,1,0,0,0 / 0,0,0,0,1,0,0 / 0,0,0,0,0,1,0] ; q = [0/0/0] $
MATRIX ; bu = <X'X>*X'y ; c = r*'<X'X>*r' ; d = r*b - q
; bc = bu - <X'X>*r'*<c>*d $
```

The second hypothesis to be tested is that the elasticities with respect to new and used cars are equal to each other, which is

\( H_0: \beta_4 = \beta_5 \).

To carry out this test using the \( F \) statistic, we can use any of the devices we used earlier. The most straightforwad would be to place the restriction in the \texttt{REGRESS} command;

```
REGRESS ; Lhs = logg ; Rhs = x ; Cls: b(4) – b(5) = 0 $ 
```

The regression results with this restriction imposed are shown below. In order to carry out the test using a t statistic, we will require
\[
t = \frac{b_4 - b_5}{\sqrt{v_{44} + v_{55} - 2v_{45}}}
\]

Where \( v_{ki} \) is the estimated covariance of \( b_k \) and \( b_l \). After computing the unrestricted regression, we can use CALC to obtain this result:

```plaintext
REGRESS : Lhs = logg ; Rhs = x$
CALC ; List : tstat = (b(4)-b(5)) / sqr(varb(4,4)+varb(5,5)-2*varb(4,5))
; tc = ttb (.975, degfrdm)
; pvalue = 2*(1-tds(tstat,degfrdm))$
```

```plaintext
+----------------------------------------------------+
| Linearly restricted regression                      |
| LHS=LOGG    Mean            =  -12.24504     |
|              Standard deviation =   .2388115     |
| WTS=none    Number of observs. =         52     |
| Model size  Parameters       =          6     |
| Degrees of freedom =         46     |
| Residuals   Sum of squares   =   .1014853     |
|              Standard error of e =   .4697022E-01 |
| Fit         R-squared        =   .9651083     |
|              Adjusted R-squared =   .9613157     |
| Model test  F[  5,    46] (prob) = 254.47 (.0000) |
| Autocorrel  Durbin-Watson Stat. =   .4411613     |
| Rho = cor[e,e(-1)] =   .7794194     |
| Restrictns. F[  1,    45] (prob) =    .02 (.8841) |
| Not using OLS or no constant. Rsqd & F may be < 0. |
| Note, with restrictions imposed, Rsqd may be < 0. |
+----------------------------------------------------+
```

```plaintext
+--------+--------------+----------------+--------+--------+----------+
|Variable| Coefficient  | Standard Error |t-ratio |P[|T|>t]| Mean of X|
|--------|--------------|----------------|--------|--------|----------|
|Constant|   -26.7523913      1.47691600   -18.114   .0000
|LOGPG   |    -.05231601       .04095506    -1.277   .2080   3.72930296
|LOGI    |    1.63099549       .15903625    10.255   .0000   9.67214751
|LOGPNC  |    -.06096105       .05689811    -1.071   .2897   4.38036654
|LOGPUC  |    -.06096105       .05689811    -1.071   .2897   4.10544883
|LOGPPT  |    -.05636622       .11703582     -.482   .6324   4.14194132
|T       |    -.01262751       .00491914    -2.567   .0137   25.5000000
```

```plaintext
+------------------------------------+
| Listed Calculator Results          |
+------------------------------------+
TSTAT   =       .146654
TC      =      2.014103
PVALUE  =       .884061
```

### 6.5.8 Test of Structural Break

The ‘Chow test’ is used to determine if the same model should apply to two (or more) subsets of the data. Formally, the test is carried out with the F statistic,

\[
F[K, n_1 + n_2 - 2K] = \frac{(SSE_{pooled} - (SSE_1 + SSE_2))/K}{(SSE_1 + SSE_2)/(n_1 + n_2 - 2K)}
\]
Where \( \text{SSE}_j \) is the sum of squared residuals from the indicated regression and \( K \) is the number of coefficients (variables plus the constant) in the model. Superficially, this requires us to compute the three regressions, then compute the statistic. We might not be interested in seeing the actual regression results for the subperiods, so we go directly to the sum of squares function using the calculator. We proceed as follows:

\[
\text{PERIOD} : 1953-2004 \$
\text{CALC} : \text{sse} = \text{Ess}(x, \text{logg}) ; k = \text{col}(x) \$
\text{PERIOD} : 1953 – 1973 \$
\text{CALC} : \text{sse1} = \text{Ess}(x, \text{logg}) ; n1 = n \$
\text{PERIOD} : 1974 – 2004 \$
\text{CALC} : \text{sse2} = \text{Ess}(x, \text{logg}) ; n2 = n \$
\text{CALC} : \text{list} ; f = \frac{(\text{sse} - (\text{sse1} + \text{sse2}))/k}{((\text{sse1} + \text{sse2})/(n1+n2-2*k))} ; fc = \text{Ftb}(.95, k, (n1+n2-2*k))$

The results shown are the sample \( F \) and the critical value from the \( F \) table. The null hypothesis of equal coefficient vectors is clearly rejected.

\[
\begin{align*}
| & \text{Listed Calculator Results} & | \\
\hline
F & = & 83.664571 \\
\text{FC} & = & 2.262304 \\
\hline
\end{align*}
\]

The second approach suggested is to use a Wald test to test the hypothesis of equal coefficients. For the structural change test, the statistic would be

\[
W = (b_1 - b_2)' [V_1 + V_2]^{-1} (b_1 - b_2).
\]

The difference in the two approaches is that the Wald statistic does not assume that the disturbance variances in the two regressions are the same. We compute the test statistic using matrix algebra then obtain the critical value from the chi squared table.

\[
\text{PERIOD} : 1953 – 1973 \$
\text{REGRESS} : \text{Lhs} = \text{logg} ; \text{Rhs} = x ; \text{quietly} \$
\text{MATRIX} : b1 = b ; v1 = \text{varb} \$
\text{PERIOD} : 1974 – 2004 \$
\text{REGRESS} : \text{Lhs} = \text{logg} ; \text{Rhs} = x ; \text{quietly} \$
\text{MATRIX} : b2 = b ; v2 = \text{varb} \$
\text{MATRIX} : \text{db} = b1 - b2 ; vdb = v1 + v2 ; \text{list} ; W = \text{db}'<vdb>db \$
\text{CALC} : \text{list} ; cstar = \text{ctb}(95, k) \$
\]

The results of the Wald test are the same as the \( F \) test. The hypothesis is decisively rejected.

\[
\text{Matrix W} \quad \text{has} \quad 1 \text{ rows and} \quad 1 \text{ columns.} \quad 1
\]

\[
\text{1|} \quad 612.92811
\]

\[
\begin{align*}
| & \text{Listed Calculator Results} & | \\
\hline
\text{Result} & = & 14.067140 \\
\hline
\end{align*}
\]
This page intentionally left blank.
Essentials of Data Management

Chapter 7

Essentials of Data Management

7.1 Introduction

The tutorial in Chapter 6 showed how to read a simple data file, how to obtain transformed variables (logs), and how to change the sample by setting the period for time series data. There are many different options for these operations in LIMDEP. This chapter will describe a few of the additional features available in the program for managing your data. The sections to follow are:

7.2: Reading and Entering Data
7.3: Computing Transformed Variables
7.4: Lists of Variables
7.5: The Current Sample of Observations
7.6: Missing Data

7.2 Reading and Entering Data

Most of the analyses of externally created data that you do with LIMDEP will involve data sets that are entered via disk files. (The alternative is internally created data that you produce using LIMDEP’s random number generators.) The READ command is provided for this purpose.

7.2.1 The Data Area

Your data are stored in an area of memory that we will refer to as the data array. The data array in the student version of LIMDEP or NLOGIT contains 1,000 rows and 99 columns. These values are preset and cannot be changed.

7.2.2 The Data Editor

For initial entry of data, LIMDEP’s data editor resembles familiar spreadsheet programs, such as Microsoft Excel. You can reach the data editor in several ways:

- Click the data editor (grid) icon on the LIMDEP toolbar,
- Double click any variable name in the project window.
- Select the menu entry Project: Data Editor.

The data editor is shown in Figure 7.1. (The chevrons next to the observation numbers indicate that these observations will be in the current sample. See Section 7.5.)
The data editor appears with an empty editing area when no variables exist. The functions of the data editor are shown in the smaller menu, which you obtain by pressing the right mouse button, displayed in Figure 7.1. The functions are:

**New Variable**

Click **New Variable** to open a dialog box that will allow you to create new variables with transformations as shown in Figure 7.2. (This is analogous to the CREATE command described in Section 7.3.) If you wish simply to type in a column of data – a variable – in the New Variable dialog box, just enter the name at the top of the box and click **OK**. This will create a new variable with a single observation equal to zero, which you will replace, and all remaining observations missing. You will then enter the data editor where you can type in the values in the familiar fashion.
The New Variable dialog box also allows you to transform existing variables after they have been entered. For example, in Figure 7.2, the dialog is being used to create a variable named \textit{logprod} using two existing variables named \textit{y} and \textit{x} and the Log and Phi (normal CDF) functions.

After you create and enter new variables, the project window is updated, and the data, themselves, are placed in the data area. Figure 7.3 illustrates.

![Data Editor and Project Window](image)

**Figure 7.3 Data Editor and Project Window**

**Import Variables**

Click Import Variables to open a dialog box that allows you to import a spreadsheet or other data file. This is analogous to the \texttt{READ} command described below.

**Set Sample**

Click Set Sample to obtain a menu of options for setting the current sample. This uses the \texttt{REJECT}, \texttt{INCLUDE} and \texttt{DATES} features discussed in Section 7.5 to set the current sample. (This option is not available until data have been read into the data area.)

**HINT:** The data editor does not automatically reset the current sample. After you enter a data set with it, the current sample is unchanged.

Data may be edited in the data editor in the usual fashion for spreadsheets. Note, however, that this means only entering new values or replacing old ones. We have not replicated the full transformation functionality in, e.g., Excel. Data may be transformed and manipulated using other features in LIMDEP. The New Variable... option, however, does provide all the features you might need to transform variables. Exit the editor simply by closing the window.
7.2.3 Reading Data Files Into LIMDEP

Data may be entered into LIMDEP from many different kinds of files. Formats include simple ASCII files, spreadsheet files written by other programs such as Excel, and binary as well as other types of files written by other statistical programs. The usual way of entering data in LIMDEP if you are not typing them in the data editor is in the form of ASCII text. An ASCII file is a text file that you can display in a word processing program. This section will describe how to read ASCII files. You will typically ‘read’ an ASCII file by instructing LIMDEP to locate the file on one of your computer’s storage drives and ‘read’ the file. The next few sections of this chapter will show you how to do that. We begin, however, with a simpler operation that does the same thing. (Section 6.3 also shows you how to read data into LIMDEP using the text editor.)

Simple Rectangular ASCII File

The simplest, basic starting point for reading data files, instead of from the text editor, is the rectangular ASCII file that contains exactly one row of variable names at the top of the file and columns of data below it, as shown in the example in Figure 7.4.

<table>
<thead>
<tr>
<th>ID</th>
<th>Year</th>
<th>Age</th>
<th>Educ</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1960</td>
<td>23</td>
<td>16</td>
</tr>
<tr>
<td>2</td>
<td>1975</td>
<td>44</td>
<td>12.5</td>
</tr>
<tr>
<td>3</td>
<td>1990</td>
<td>14</td>
<td>11.5</td>
</tr>
<tr>
<td>4</td>
<td>1993</td>
<td>missing</td>
<td>20</td>
</tr>
</tbody>
</table>

Figure 7.4 Sample Data File

The sample data set shown illustrates several degrees of flexibility.

- Variable names in a file may be separated by spaces and/or commas.
- Names need not be capitalized in the file. LIMDEP will capitalize them as they are read.
- The numbers need not be lined up in neat columns in a data file.
- Values in the data set may be separated by spaces, tabs, and/or commas.
- Missing values in a data set may be indicated by anything that is not a number. (But, they must be indicated by something. A blank is never understood to be a missing value.)

To read a data file of this form, you need only tell LIMDEP where it is. The command to read this file is

```
READ ; File = … < the name of the file > $
```

The READ command may be submitted from the text editing window or in the command bar at the top of the screen. There are two other ways to import a data file of this form:

- Use Project:Import Variables to open the Import dialog box, as shown in Figure 7.5. Select All Files (*.*) in the file types, then locate and select your data file and click Open.
- In the data editor, select Import Variables from the menu invoked by clicking the right mouse button. (See Figure 7.1.) This opens the same Import dialog box described above.
When you read a file of this type, LIMDEP determines the number of variables to be read by counting the number of names that appear in the first line of the file. The number of observations in the file is determined by reading until the end of the file is reached.

**Obtaining the Path to a File**

The preceding application is one of several situations in which you will need to specify the full path to a file. Sometimes this is hard to locate. You can obtain the full path to a file by using Insert:File Path. For example, where is the file “TableF1-1.txt” that is selected in Figure 7.5? Step one is to make sure that your text editing window is active. The file path will be inserted where the cursor is. Insert:File Path brings up exactly the dialog box shown in Figure 7.5 except that the banner title will be Insert File Path instead of Import. When you click Open, the full path to the file will be placed in double quotes in the text editor. The result is shown in Figure 7.6.
7.2.4 General ASCII Files

The more general command for reading an ASCII data file is

```
READ ; Nvar = number of variables
; Nobs = number of observations
; Names = names for the variables
; File = the full file name, including path $
```

(Note that the preceding shows the ‘default’ form of this command.) This assumes that the file is an ASCII file (not a spreadsheet) with numbers arranged in rows, separated by blanks, tabs, and/or commas. Numbers in the file need not be neatly arranged vertically. If there are missing values, there must be placeholders for them; since blanks just separate values, they cannot be interpreted as missing data. Use as many as lines as needed for each observation to supply all of the values. For example, for the following data in a file,

```
1 2 5
3 4 6
2 5 4
3 6 7
```

you might use

```
READ ; File = \Project\SMALL.DAT ; Nobs = 4 ; Nvar = 3 ; Names = x,y,z $
```

There are several optional features, and a number of other different types of data files you can use.

**HINT**: If you do not know the exact number of observations in your data set, give Nobs a number that you are sure will be larger than the actual value. *LIMDEP* will just read to the bottom of the file and adjust the number of observations appropriately.

**Variable Names**

The normal way to enter variable names is in the command, as in the example above,

```
; Names = name_1,...,name_nvar
```

The following name conventions apply: names may have up to eight characters, must begin with a letter, and must be composed from only letters, numbers, and the underscore character. Remember that names are always converted to upper case. Reserved names are listed in Section 4.3.

**Names in the Data File**

You may find it convenient to make the variable names a part of the data set instead of the *READ* command. (Note that this is the basic case described in the tutorial in the previous chapter.) To do so, add the specification

```
; Names = n
```
to your **READ** command, where \( n \) is the number of lines you need to list the names. Then, at the absolute beginning of the data set, include exactly \( n \) lines of 80 or fewer characters containing the variable names, separated by any number of spaces and/or commas. (For appearance’s sake, you might, for example, position the names above the variables.)

The following reads a data set containing three observations on four variables:

```
READ ; Nobs = 3 ; Nvar = 4 ; Names = 2 ; File ="... MACRO.DAT “$
```

The data set could be

<table>
<thead>
<tr>
<th>Year</th>
<th>Consmptn</th>
<th>GNP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Govt</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1961</td>
<td>831.25</td>
<td>996.19</td>
</tr>
<tr>
<td>1962</td>
<td>866.95</td>
<td>1024.82</td>
</tr>
<tr>
<td>1963</td>
<td>904.44</td>
<td>1041.03</td>
</tr>
</tbody>
</table>

**Observation Labels**

The data file below contains labels for the individual observations as well as the names of the variables. You may read a file with observation labels with the following format:

```
READ ; File = ... filename... ; Nobs = ... ; Nvar = ...
; Labels = the column in the data file that contains the labels $
```

The labels column is an extra column in the data. It is not a variable. For the file below, you would use

```
READ ; ... ; Nvar = 4 ; Nobs = 25 ; Names = 1 ; Labels = 1 $
```

This indicates that the labels are in the first column, which is probably typical.

<table>
<thead>
<tr>
<th>State</th>
<th>ValueAdd</th>
<th>Capital</th>
<th>Labor</th>
<th>NFirm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alabama</td>
<td>126.148</td>
<td>3.804</td>
<td>31.551</td>
<td>68</td>
</tr>
<tr>
<td>California</td>
<td>3201.486</td>
<td>185.446</td>
<td>452.844</td>
<td>1372</td>
</tr>
<tr>
<td>Connecticut</td>
<td>690.670</td>
<td>39.712</td>
<td>124.074</td>
<td>154</td>
</tr>
<tr>
<td>Florida</td>
<td>56.296</td>
<td>6.547</td>
<td>19.181</td>
<td>292</td>
</tr>
<tr>
<td>Georgia</td>
<td>304.531</td>
<td>11.530</td>
<td>45.534</td>
<td>71</td>
</tr>
<tr>
<td>... (20 more observations)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

You must include a name for the labels. Note that the ‘State’ name is used for the labels, but not for the data.

**7.2.5 Reading a Spreadsheet File**

You can read worksheet files such as those created by *Lotus* (.WKx) or Microsoft *Excel* (.XLS) directly into LIMDEP without conversion. The command is simply

```
READ ; File = name ; Format = WKS $
```
or

```
READ ; File = name ; Format = XLS $
```

All of the information needed to set up the data is contained within the file. However, if the columns in your worksheet were created by formulas within the spreadsheet program, rather than
being typed in initially, then \textit{LIMDEP} will not find the transformed data. The reason is that the .XLS file does not contain the data, themselves, but merely the formula for recreating the data. Since we have not replicated the spreadsheet program inside \textit{LIMDEP}, it is not possible to redo the transformation.

\textbf{NOTE:} Please note the caution about Excel 2007 in Section 6.3.

\section*{7.2.6 Missing Values in Data Files}

\textit{LIMDEP} will catch nonnumeric or missing data codes in most types of data sets. In general, any value not readable \textit{as a number} is considered a missing value and given the value -999.

\textbf{NOTE: In all settings, -999 is \textit{LIMDEP}'s internal missing data code.}

Some things to remember about missing data are:
- A blank in a data file is normally not a missing value; it is just a blank. Since a data file can be arranged any way you want, \textit{LIMDEP} has no way of knowing that a blank is supposed to be interpreted as a missing value. But, all other nonnumeric, nonblank entries are treated as missing. This includes \textit{SAS}'s '.' character, the word ‘missing,’ or any other code you care to use.
- There will be occasions when \textit{LIMDEP} claims it found missing values when you did not think there were any. The cause is usually an error in your \textit{READ} command. For example, if you have a set of names at the top of your data file and you forget to warn the program to expect them, they will be read as missing values rather than as variable names.
- After a data set is \textit{READ}, you are given a count of the variables then existing and a summary of any missing values found.

When a data set contains missing values, you must indicate this in some way at the time the data are read. How you do this depends on the type of file you are reading:

- **Worksheet file from a spreadsheet program:** Blank cells in a worksheet file are sufficient to indicate missing data. It is not necessary to put any alphabetic indicator in the cell.

- **Unformatted ASCII file:** Any nonnumeric data in the field, such as the word ‘missing’ will suffice. Alternatively, a simple period surrounded by blanks will suffice. Note that in such a file, a blank will not be read as missing, since blanks just separate numbers in the data file.

The internal code for a missing datum is -999. You may use this numeric value in any type of file to indicate a missing value. Upon reading the data, \textit{LIMDEP} immediately converts any missing data encountered to the numeric value -999.
7.3 Computing Transformed Variables

You will usually need to transform your data, for example to obtain logarithms, differences, or any number of other possibilities. LIMDEP provides all of the algebraic transformations you are likely to need with the CREATE command. It is often useful to recode a continuous variable into discrete values or to combine discrete values into a smaller number of groups, for example to prepare data for contingency tables. The RECODE command is provided for this purpose. You can use SORT to arrange one or more variables in ascending or descending order.

7.3.1 The CREATE Command

The CREATE command is used to modify existing variables or compute new ones. The essential syntax of the command is

```
CREATE ; name = expression ; name = expression ; ... $
```

You may also enter your command in a dialog box, as shown in Figure 7.10. The dialog box is invoked by selecting Project: New/Variable or by going to the data editor and pressing the right mouse button which will bring up a menu that includes New Variable. You may now enter the name for the new or transformed variable in the Name window. If you click OK at this point without entering an expression for the variable, the new variable is created, and the first observation is set to zero. Remaining observations are treated as missing. You may enter an expression for the new or transformed variable in the Expression window.

![Figure 7.10 New Variable Dialog Box](image)
A **CREATE** command operates on the ‘current sample.’ (See Section 7.5.) If this is a subset of the data, remaining observations will not be changed. If you are creating a new variable for the subset of observations, remaining observations will be undefined (missing). You can override this feature by using

\[ \text{CREATE ; Fill ; ... the rest of the command} \]

in your command. With this additional setting, the transformations listed will be applied to all observations in the data set, whether in the current sample or not. This is the **Data fill** option that appears at the bottom center of the dialog box in Figure 7.10.

### Algebraic Transformations

An algebraic transformation is of the form: \( \text{name} = \text{expression} \). *Name* is the name of a variable. It may be an existing variable or a new variable. *Name* may have been read in or previously created.

The expression can be any algebraic transformation of any complexity. You may nest parentheses, functions, and operations to any level. Functions that may appear in expressions are listed in Section 7.3.3. The operators that may appear in **CREATE** commands are the standard ones, +, −, *, and / for addition, subtraction, multiplication, and division, as well as the special operators listed below:

- \(^\) = raise to the power; \( a^b = a^b \)
- @ = Box-Cox transformation; \( a@b = (a^b - 1)/b \) or \( \log a \) if \( b = 0 \) and \( a > 0 \)
- ! = maximum; \( a!b = \max(a,b) \)
  (The maximum of a string of operands is obtained just by writing the set separated by !s. For example, 5!3!6!0!1 = 6.)
- ~ = minimum; \( a~b = \min(a,b) \)
- % = percentage change; \( a\%b = 100(a/b - 1) \) E.g., 5\%4 = 25

The following operators create binary variables:

- > = binary variable; \( a>b = 1 \) if \( a > b \) and 0 else.
- >= = binary variable; \( a\geq b = 1 \) if \( a \geq b \) and 0 else.
- < = binary variable; \( a<b = 1 \) if \( a < b \) and 0 else.
- <= = binary variable; \( a\leq b = 1 \) if \( a \leq b \) and 0 else.
- = = binary variable; \( a=b = 1 \) if \( a = b \) and 0 else.
- # = binary variable; \( a\#b = 1 \) if \( a \) is not equal to \( b \).

For example,

\[
\text{CREATE ; a = x > 0 * Phi(y)} \quad \text{creates } a \text{ equal to } \Phi(y) \text{ if } x \text{ is positive and 0 else}
\]

\[
\text{; p = z > 0} \quad \text{creates } p = 1 \text{ if } z \text{ is positive and 0 otherwise}
\]

\[
\text{; zeq1 = z = 1 S} \quad \text{equals 1 if } z \text{ equals 1 and 0 otherwise.}
\]

To avoid ambiguity, it is often useful to enclose these operations in parentheses, as in

\[
\text{CREATE ; a = (x = 1) * Phi(z) S}
\]
This set of tools can be used in place of conditional commands, and sometimes provides a convenient way to make conditional commands. For example, ‘and’ conditions result from products of these relational operators. Thus,

\[
\text{CREATE } \quad ; v = (x \geq 8) \times (x \leq 15) \times \log(q)
\]

creates \( v \) equal to the log of \( q \) if \( x \) is greater than or equal to 8 and less than or equal to 15. You can also produce an ‘or’ condition using addition, though the conditional command construction may be more convenient. For example:

\[
\text{CREATE } \quad ; v = \left( \left( \frac{(x = 8) + (x = 15)}{0} \right) \right) \times \log(q)
\]

does the transformation if \( x \) equals 8 or 15.

The following algebraic order of precedence is used to evaluate expressions:

- First: functions, such as \( \log(.) \) are evaluated.
- Second: \(^\wedge\) and @, which have equal precedence are computed.
- Third: \( *, /, !, ~, %, >, >=, <, <=, =, # \) are computed.
- The special operators, !, %, etc. are evaluated from left to right with the same precedence as \( * \) and \( / \). Thus, for example, \( y * x > 0 \) equals 1 if \( y \times x \) is greater than 0 and equals 0 otherwise. It will usually be useful to use parentheses to avoid ambiguities in these calculations.
- Fourth: \( + \) and \( - \) (addition and subtraction) are computed.

**NOTE:** \textit{LIMDEP} does not give the unary minus highest precedence. The expression \(-x^2\) evaluates to the negative of the square of \( x \) (which would be negative) not the square of negative \( x \) (which would be positive). This is the current standard in software, but it is not universal.

You may use as many levels of parentheses as necessary in order to group items in an expression or to change the order of evaluation. For example,

\[
\text{CREATE } \quad ; \text{ma} = (pz + pz[-1] + pz[-2] + pz[-3]) / 4
\]

computes a moving average of a current and three lagged values. Parentheses may also be nested to any depth.

\[
\text{CREATE } \quad ; \text{ratio} = \frac{((x + y)^2 - (a + c)^2)^2}{((a + x)\times(c + y))}
\]

is a valid command which computes \( \text{ratio} = \frac{((x + y)^2 - (a + c)^2)^2}{(a + x)(c + y)} \).

You may also nest functions. For a few examples, the following functions are used to invert certain probability distributions:
Gompertz: \[ \text{CREATE} \ ; \ t = \log(1 - w \cdot \log(a)/p) / w \]
Weibull: \[ \text{CREATE} \ ; \ t = (-\log(a))^{(1/p)} / w \]
Normal: \[ \text{CREATE} \ ; \ t = \exp(-\ln(p(a)/p)) / w \]
Logistic: \[ \text{CREATE} \ ; \ t = ((1 - a) / a)^{(1/p)} / w \]

Functions may be nested to any depth, and expressions may appear in the parentheses of a function. Consider, for example, the following which creates the terms in the log likelihood function for a tobit model

\[ \text{CREATE} \ ; \ \text{loglik} = (1 - d) \cdot \log(\Phi(-x'b/sigma)) + d \cdot \log((1/sigma) \cdot N01((y-x'b)/sigma)) \]

**Cautions:**

- Any transformation that involves a missing value (-999) at any point returns a missing value.
- It is unlikely to be necessary, but if you should require expressions in the parameter list of a two parameter function, put them in parentheses. The Trn function which computes trend variables is such a function. Thus,

\[ \text{CREATE} \ ; \ \text{trend} = \text{Trn}(a + b'x, \text{step}) \]

would confuse the compiler. Instead, you should use

\[ \text{CREATE} \ ; \ \text{trend} = \text{Trn}(a + b'x, \text{step}) \]

- Many operations allow you to access particular observations of a variable by using an observation subscript enclosed in parentheses. If you will be using this construction, you must avoid variable names which are the same as the function names listed in Section 7.3.3. For example, if you have a variable named \textit{phi}, then \( \Phi(1) \) could be the first observation on \textit{phi} or the standard normal CDF evaluated at 1.0. (\texttt{CREATE} will translate it as the latter.) Function names all have three letters. You should examine the list given in Section 7.3.3.

Variables may appear on both sides of the equals sign as long as they already exist, and transformations may be grouped in a single command. In a multiple \texttt{CREATE} command, later transformations may make use of variables created in earlier ones. For example,

\[ \text{CREATE} \ ; \ \text{sam} = x1 \cdot x2 \]
\[ \text{bob} = x2 + x3 \]
\[ \text{this} = \text{sam} \cdot \text{bob} \]
\[ \text{that} = \log(\text{this}) \]
\[ \text{that} = 1 / \text{that} \]

is the same as five consecutive \texttt{CREATE} commands. You should write your transformations so that they are as ‘self documenting’ as possible – that is, so that they are as easy to understand as possible.
7.3.2 Conditional Transformations

Any transformation may be made conditional. The essential format is

CREATE ; If ( logical expression ) name = expression $

Logical expressions are any desired expressions that provide the condition for the transformation to be carried out. They may include any number of levels of parentheses and may involve mathematical expressions of any complexity involving variables, lagged variables of the form name[lag], named scalars, matrix or vector elements, and literal numbers. The operators are the same as above with a few exceptions: The ones that may be used are the math and relational operators: +, -, *, /, ^, >, >=, <, <=, =, #. The special operators, @, !, %, and ~ are not used here.

NOTE: Logical expressions may not involve functions such as Log, Exp, etc.

Concatenation operators which can be used for transformations are & for ‘and,’ and | for ‘or.’ A simple example might be: CREATE ; If ( x > 0 ) ... expression $  For a more complex example, we compute an expression for observations which are not inside a ball of unit radius.

CREATE ; If ( x1^2 + x2^2 + x2^2 >= 1 ) ... expression... $

The hierarchy of operations is ^, (*, /, +,-), (>==,<==,=,=,#), &, |. Operators in parentheses have equal precedence and are evaluated from left to right. When in doubt, add parentheses. There is essentially no limit to the number of levels of parentheses. (They can be nested to about 20 levels.)

7.3.3 Transformations Involving Missing Values

NOTE: Any mathematical expression that involves a missing value produces a missing value as the result.

Any transformation that requires a value which turns out to be a cell containing missing data will return a missing value, not 0. Thus, if you compute \( y = \log(x) \), and some values of \( x \) are missing, the corresponding values of \( y \) will be also.

When computing a column of predictions, LIMDEP returns a missing value for any observations for which any of the variables needed to compute the prediction are missing, even if the variable which will contain the predictions already exists at the time. This results because when you request a model to produce a set of predictions, LIMDEP begins the process by ‘clearing’ the column in the data area where it will store the predictions. Data areas are cleared by filling them with the missing value code.
7.3.4 CREATE Functions

The expressions in CREATE may involve the following functions:

**Common Algebraic Functions**

Log(x)  = natural logarithm,
Exp(x)  = exponent,
Abs(x)  = absolute value,
Sqr(x)  = square root,
Sin(x)  = sine,
Rsn(x)  = arcsine (operand between -1 and 1),
Cos(x)  = cosine,
Rcs(x)  = arccosine (operand between -1 and 1),
Tan(x)  = tangent,
Ath(x)  = hyperbolic arctangent = \( \frac{1}{2} \log((1+x)/(1-x)) \), -1 < x < 1,
Ati(x)  = inverse hyperbolic arctangent = \[ \frac{\exp(2x)-1}{\exp(2x)+1} \],
Gma(x)  = gamma function = \( (x-1)! \) if x is an integer,
Psi(x)  = digamma = log-derivative of gamma function = \( \Gamma'/\Gamma = \Psi(x) \),
Psp(x)  = trigamma = log-2nd derivative of gamma = \( \frac{\Gamma\Gamma'' - \Gamma'}{\Gamma^2} = \Psi'(x) \),
Lgm(x)  = log of gamma function (returned for Gma if x > 50),
Sgn(x)  = sign function = -1,0,1 for x <, =, > 0,
Fix(x)  = round to nearest integer,
Int(x)  = integer part of operand.

**Univariate Normal and logistic Distributions**

Phi(x)  = CDF of standard normal,
N01(x)  = PDF of standard normal,
Lgf(x)  = log of standard normal PDF = -.5(\log(2\pi + x^2)) = Log(N01(x)),
Lmm(x)  = -N01/Phi = \( E[x | x < \text{operand}] \), x ~ N(0,1),
Lmp(x)  = N01/(1-Phi) = \( E[x | x > \text{operand}] \),
Lmd(x,z) = (z-1)Lmp(x) - zLmm(x) where z = 0/1 (selectivity variable),
Tvm(x)  = [1 - Lmm(Lmm+z)] = Var[\( x | x < \text{operand} \)],
Tvp(x)  = [1 - Lmp(Lmp+z)] = Var[\( x | x > \text{operand} \)],
Tvr(x,z) = (1-z)Tvm(x) + zTvp(x) where z = 0/1 (selected variance),
Inp(x)  = inverse normal CDF,
Inf(x)  = inverse normal PDF (operand is CDF, returns density),
Lgt(x)  = logit = \log(z/(1-z)),
Lgp(x)  = logistic CDF = \exp(x)/(1 + \exp(x)),
Lgd(x)  = logistic density = Lgp(1-Lgp).

**Trends and Seasonal Dummy Variables**

Trn(x1,x2) = trend = x1+(i-1)x2 where i = observation number,
Ind(i1,i2) = 1 if i1 ≤ observation number ≤ i2, 0 else,
Dmy(p,i1) = 1 for each pth observation beginning with i1, 0 else.

The Dmy function is used to create seasonal dummy variables. The Ind function operates on specific observations, as in
CREATE ; eighties = Ind (22,31) $

If your data are time series and have been identified as such with the DATES command (see Chapter R7), then you may use dates instead of observation numbers in the Ind function, as in

CREATE ; eighties = Ind (1980.1,1989.4) $

**NOTE:** The Ind function is oblivious to centuries. You must provide four digit years to this function, so there is no ambiguity about 19xx vs. 20xx.

The trend function, Trn is used to create equally spaced sequences of values, such as 1,2,3,..., which is Trn(1,1). There are two additional variants used primarily with panel data. These are discussed in Chapter R6.

**Leads and Lags**

You can use a lagged or leaded variable with the operand

variable [ n ] = observation on the variable n periods prior or ahead.

The use of square brackets is mandatory; ‘n’ is the desired lag or lead. If n is negative, the variable is lagged; if it is positive, it is leaded. For example, a familiar transformation, Nerlove’s ‘universal filter’ is \((1 - .75L)^2\) where \(L\) is the lag operator. This would be

CREATE ; filterx = x - 1.5 * x[-1] + .5625 * x[-2] $

A value of -999 is returned for the operand whenever the value would be out of the range of the current sample. For example, in the above command, \(filterx\) would equal -999 for the first two observations. You can change this default value to something else, like zero, with

CREATE ; [LAG] = the desired value $

For example,

CREATE ; [LAG] = 0 $

would change the default value for noncomputable lags to zero. This must be used in isolation, not as part of some other command.

If you use lags or leads, you should modify the applicable sample accordingly when you use the data for estimation. LIMDEP makes no internal note of the fact that one variable is a lagged value of another one. It just fills the missing values at the beginning of the sample with -999s at the time it is created.

Moving average and autoregressive sequences can easily be constructed using CREATE, but you must be careful to set up the initial conditions and the rest of the sequence separately. Also, remember that CREATE does not reach beyond the current sample to get observations. A special read-only variable named _obsno (note the leading underscore) is provided for creating recursions. Consider computing the (infinite) moving average series

\[ y_t = x_t + \theta x_{t-1} + \theta^2 x_{t-2} + \ldots + \theta^{t-1} x_{t-1}. \]
To do the computation, we would use the autoregressive form, $y_t = x_t + \theta y_{t-1}$ with $y_1 = 0$. The following could be used:

```
CREATE ; If(_obsno = 1) y = 0
         ; (Else)          y = x + theta * y[-1] $,
```

Second, consider generating a random sample from the sequence $y_t = \theta y_{t-1} + e_t$, where $e_t \sim \mathcal{N}[0,1]$. Simply using `CREATE ; y = theta*y[-1] + Rnn(0,1)` will not work, since, once again, the sequence must be started somewhere. But, you could use the following:

```
CREATE ; If(_obsno = 1) y = Rnn(0,1) / Sqr(1 - theta^2)
         ; (Else)          y = theta * y[-1] + Rnn(0,1) $,
```

**Matrix Function**

A transformation based on matrix algebra is used to create linear forms with the data. Linear combinations of variables are obtained with:

```
CREATE ; name = b’x $,
```

where $x$ is a namelist of variables (see Section 7.5) and $b$ is any vector with the same number of elements. It creates the vector of values from the linear combination of the variables in the namelist with coefficients in the row or column matrix. Dot products may also be used with other transformations. For example,

```
CREATE ; bx12 = x1’b1 / x2’b2 ; p = Phi(x’b/s) $,
```

(The order is not mandatory. $d’z$ is the same as $z’d$.) Also, if you need this construction, a dot product may be used for two vectors or two namelists. In the latter case, the result is the sum of squares of the variables.
7.3.5 Expanding a Categorical Variable into a Set of Dummy Variables

It is often useful to transform a categorical variable into a set of dummy variables. For example, a variable, educ, might take values 1, 2, 3, and 4, for less than high school, high school, college, post graduate. For purposes of specifying a model based on this variable, one would normally expand it into four dummy variables, say underhs, hs, college, postgrad. This can easily be done with a set of CREATE commands, involving, for example, hs = (educ=2) and so on. LIMDEP provides a single function for this purpose, that simplifies the process and also provides some additional flexibility. The categorical variable is assumed to take values 1,2,...,C. The command

\[
\text{CREATE } \text{ ; Expand (variable) = name for category 1, ... name for category C }
\]

does the following:

- A new dummy variable is created for each category. (If the variable to be created already exists, it is overwritten).
- A namelist is created which contains the names of the new variables. The name for the namelist is formed by appending an underscore both before and after up to six characters of the original name of the variable.
- A tabulation of the transformation is produced in the output window.

The example suggested earlier might be simulated as follows, where the commands and the resulting output are both shown:

\[
\begin{align*}
\text{CREATE } & \text{ ; educ = Rnd(4) } \\
\text{CREATE } & \text{ ; Expand (educ) = underhs, hs, college, postgrad } \\
\end{align*}
\]

EDUC was expanded as _EDUC_.
Largest value = 4. 4 New variables were created.
Category
1 New variable = UNDERHS Frequency= 28
2 New variable = HS Frequency= 22
3 New variable = COLLEGE Frequency= 30
4 New variable = POSTGRAD Frequency= 20
Note, this is a complete set of dummy variables. If you use this set in a regression, drop the constant.

As noted, the transformation begins with the value 1. Values below 1 are not transformed and no new variable is created for the missing category. Also, the transformation does not collapse or compress the variable. If you have empty categories in the valid range of values, the variable will simply always take the value 0.0. Thus, if educ had been coded 2, 4, 6, 8, then the results of the transformation might have appeared as shown below

EDUC was expanded as _EDUC_.
Largest value = 8. 4 New variables were created.
Category
1 New variable = UNDERHS Frequency= 0 <---- !
2 New variable = HS Frequency= 23
3 New variable = COLLEGE Frequency= 0 <---- !
4 New variable = POSTGRAD Frequency= 29
5 New variable = EDUC05 Frequency= 0 <---- !
6 New variable = EDUC06 Frequency= 22
7 New variable = EDUC07 Frequency= 0 <---- !
8 New variable = EDUC08 Frequency= 26
Note, this is a complete set of dummy variables. If you use this set in a regression, drop the constant.
Things to note:

- The empty cells are flagged in the listing, but the variable is created anyway.
- If your list of names is not long enough, the remaining names are built up from the original variable name and the category value.
- The program warns you that this has computed a complete set of dummy variables. If you use this set of variables in a regression or other model, you should not include an overall constant term in the model because that would cause perfect collinearity – the ‘dummy variable trap.’ Thus, a model which contained both one and _educ_ would contain five variables that are perfectly collinear.

You may want to avoid the last of these without having to choose one of the variables to omit from the set. You can direct the transformation to drop one of the categories by adding ‘,0’ after the variable name in the parentheses.

\[
\text{CREATE } \quad \text{; Expand (variable,0) = list of names }$
\]

For our previous example, this modification would change the results as follows:

\[
\text{CREATE } \quad \text{; Expand (educ,0) = underhs, hs, college, postgrad}$
\]

EDUC was expanded as _EDUC_ .
Largest value =  4.  0 New variables were created.
Category
1  New variable = UNDERHS Frequency=  27
2  New variable = HS Frequency=  26
3  New variable = COLLEGE Frequency=  21
Note, the last category was not expanded. You may use this namelist as is in a regression with a constant.

The note at the end of the listing reminds you of the calculations done. The last category is the one dropped. (Note that ‘0 new variables were created.’ The reason is that these variables already existed after our earlier example.)

Finally, the list of names for the new variables is optional. If it is omitted, names are built up as in the second example above. Continuing the example, we might have

\[
\text{CREATE } \quad \text{; educ = Rnd(4)}$
\]

\[
\text{CREATE } \quad \text{; Expand (educ)}$
\]

EDUC was expanded as _EDUC_ .
Largest value =  4.  4 New variables were created.
Category
1  New variable = EDUC01 Frequency=  28
2  New variable = EDUC02 Frequency=  22
3  New variable = EDUC03 Frequency=  30
4  New variable = EDUC04 Frequency=  20
Note, this is a complete set of dummy variables. If you use this set in a regression, drop the constant.

\[\text{NOTE: This transformation will refuse to create more than 100 variables. If it reaches this limit, you have probably tried to transform the wrong variable. Thus, the variable must be coded 1,2,..., up to 99.}\]
### 7.3.6 Random Number Generators

There are numerous transformations which draw samples using *LIMDEP*’s random number generators. The basic generator is the one which will draw a sample from a continuous uniform distribution in the indicated range

```plaintext
CREATE ; name = Rnu (lower limit, upper limit) $
```

and the one which will create a variable containing a sample from the indicated normal distribution,

```plaintext
CREATE ; name = Rnn (mean, standard deviation) $
```

The sample is placed with the observations in the current sample. (Note how we used this feature in the examples in Sections 3.3 and 3.4.)

Random draws may also appear anywhere in an expression as operands whose values are random draws from the specified distribution. For example, a random sample from a chi squared distribution with one degree of freedom could be drawn with

```plaintext
CREATE ; name = Rnn(0,1) ^ 2 $
```

Random samples can be made part of any other transformation. For example, the following shows how to create a random sample from a regression model in which the assumptions of the classical model are met exactly:

```plaintext
CREATE ; x1 = Rnu(10,10)
; x2 = Rnn(16,10)
; y   = 100 + 1.5 * x1 + 3.1 * x2 + Rnn(0,50) $
```

The regression of \( y \) on \( x_1 \) and \( x_2 \) would produce estimates of \( \beta_1 = 100, \beta_2 = 1.5, \) and \( \beta_3 = 3.1. \)

In addition to the \( \text{Rnn}(m,s) \) (normal with mean \( m \) and standard deviation \( s \)) and \( \text{Rnu}(l,u) \) (continuous uniform between \( l \) and \( u \)), you can generate random samples from continuous and discrete.

#### Random Samples from Continuous Distributions

- \( \text{Rng}(m,s) \) = lognormal with parameters \( m \) and \( s \),
- \( \text{Rnt}(n) \) = \( t \) with \( n \) degrees of freedom,
- \( \text{Rnx}(d) \) = chi squared with \( d \) degrees of freedom,
- \( \text{Rnf}(n,d) \) = \( F \) with \( n \) numerator and \( d \) denominator degrees of freedom,
- \( \text{Rne}(q) \) = exponential with mean \( q \),
- \( \text{Rnw}(a,c) \) = Weibull with location \( a \) and scale \( c \). If \( c = 1 \), use \( \text{Rnw}(a) \).
- \( \text{Rnh}(a,c) \) = Gumbel (extreme value) with location \( a \), scale \( c \). If \( c = 1 \), use \( \text{Rnh}(a) \).
- \( \text{Rni}(a,c) \) = gamma with scale \( a \) and shape \( c \). If \( a = 1 \), use \( \text{Rni}(c) \).
- \( \text{Rna}(a,b) \) = beta with parameters \( a \) and \( b \),
- \( \text{Rnl}(0) \) = logistic,
- \( \text{Rnc}(0) \) = Cauchy.
Random Samples from Discrete Distributions

\[ \text{Rnp}(q) = \text{Poisson with mean } q, \]
\[ \text{Rnd}(n) = \text{discrete uniform, } x=1,...,n, \]
\[ \text{Rnb}(n,p) = \text{binomial, } n \text{ trials, probability } p, \]
\[ \text{Rnm}(p) = \text{geometric with success probability } p. \]

For sampling from the binomial distribution, the limits on \( n \) and \( p \) are \( n\log(p) \) and \( n\log(1-p) \) must both be greater than -264 to avoid numerical overflow errors. You must provide the 'a' in the Weibull and Gumbel and the '0', logistic, and Cauchy functions. You may also sample from the truncated standard normal distribution. Two formats are

\[ \text{Rnr}(\text{lower}) = \text{sample from the distribution truncated to the left at 'lower',} \]
\[ \text{Rnr}(\text{lower,upper}) = \text{distribution with both tails truncated.} \]

E.g., Rnr(.5) samples observations greater than or equal to .5

Parameters of all requests for random numbers are checked for validity. For the truncated normal, you must have

\[ lower \leq 1.5, upper \geq -1.5, upper - lower \geq .5 \]

If 'upper' is not provided, it is taken as \( +\infty \). If you need upper truncation, a transformation which will produce the desired result is \( -\text{Rnr}(-\text{lower}). \)

The parameters of any random number generator can be variables, other functions, or expressions, as well. For example, you might simulate draws from a Poisson regression model with

\[
\text{CREATE } ; \ x1 = \text{Rnn}(0,1) \\
; \ x2 = \text{Rnu}(0,1) \\
; \ y = \text{Rnp}(\text{Exp (.2 + .3 } * \ x1 - .05 * \ x2 )) $
\]

Setting the Seed for the Random Number Generator

To reset the seed for the random number generator (the same one is used for all distributions), use the command

\[
\text{CALC } ; \text{Ran (seed)} $
\]

In this fashion, you can replicate a sample from one session to the next. Use a large (e.g., seven digit) odd number for the seed. Using this device will allow you to draw the same string of pseudo-random numbers more than once.
7.4 Lists of Variables

As part of estimation, it is necessary to define two sets of information, the variables to be used and the observations. *LIMDEP*’s data handling and estimation programs are written to handle large numbers of variables with simple, short commands. Two methods are provided to reduce the amount of typing involved in giving a list of names, the NAMELIST and a wildcard character. The NAMELIST feature is used as follows:

```
NAMELIST ; name = the list of variables names $
```

### 7.4.1 Lists of Variables in Model Commands

Lists of variables are used in every model estimation command and a large number of other commands, such as WRITE. For example, nearly all model commands are of the form

```
MODEL COMMAND ; Lhs = a variable
 ; Rhs = a list of variables
 ; Rh2 = a list of variables $
```

Each of the lists may, in principle, have 150 or more names in it. As such, some shorthands will be essential.

One simple shorthand for lists of variable names is the wildcard character, ‘*.’ You may use the ‘*’ character to stand for lists of variables in any variable list. There are three forms:

- * stands for all variables.

```
LIST ; * $
DELETE ; * $
```

- `aaaa*` stands for all variables whose names begin with the indicated characters, any number from one to seven. For example: If you have variables `x1`, `x2`, `xa`, `xxx`, `xxy`,

```
x* = all five variables,
xx* = xxx and xxy,
```

then the following command requests simultaneous scatter plots of all variables whose names begin with `x`,

```
SPLOT ; Rhs = x* $
```

- `*aaaa` stands for variables whose names end with the indicated characters. For example, if you have `xa`, `ya`, and, `y`,

```
REGRESS ; Lhs = y ; Rhs = one, *a $
```

regresses `y` on `one`, `xa`, and `ya`.  

"
7.4.2 Namelists

The wildcard character described above will save some typing. But, namelists will usually be a more efficient approach to specifying a list of names. The NAMELIST command is used to define a single name which will be synonymous with a group of variables. It can be used at any time and applies to the entire set of variables currently in the data array, regardless of how they got there. (Variables are placed in the data array with many commands including READ, CREATE, MATRIX, and any of the model commands.) The form of the command is

\[
\text{NAMELIST } \; \text{name} = \text{list of variable names}
\]

Several namelists may be defined with the same NAMELIST command by separating the definitions with semicolons, e.g.,

\[
\text{NAMELIST } \; \text{w1} = x1, x2 \; \text{w2} = x3, x4, x5
\]

The lists of variables defined by separate namelists may have names in common. For example,

\[
\text{NAMELIST } \; \text{w1} = x1, x2 \; \text{w2} = x2, x3
\]

By double clicking or right clicking the name of a namelist in the project window, you can enter an editor that allows easy modification of namelists. See Figure 7.13 for the setup.

![Figure 7.13 Editing a Namelist](image_url)
You can also define new namelists with the **New Namelist** editor. There are several ways to reach this editor:

- Select **New/Namelist** from the **Project** menu,
- Select **Item into Project/Namelist** from the **Insert** menu,
- Right click the **Namelists** header in the project window, and select **New Namelist**.

All these will invoke the dialog box shown in Figure 7.14.

![New Namelist Dialog Box](image)

### 7.4.3 Using Namelists

Namelists will have many uses as you use **LIMDEP** to analyze your data. Consider the example,

```
NAMELIST ; job = butcher, baker, cnndlmakr
    ; place = north, south, east, west
    ; person = job, place, income $
```

Note that in the example, the namelist `person` will contain eight variables, as the other two namelists are expanded and included with the eighth variable, `income`.

Your primary use of the **NAMELIST** command will be for defining variable lists for the estimation commands. However, a namelist may be used at any point at which a list of variable names is called for. Some other applications are in defining data matrices for the **MATRIX** command and in several **CALCULATE** and **CREATE** commands. Some examples:

```
NAMELIST ; xlist = x1, x2, x3, x4 $
REGRESS ; Lhs = y ; Rhs = one, xlist $
CALC ; Col(xlist) $ How many variables in list?
WRITE ; xlist, y1, y2, y3 ; File = DATA.DAT $
MATRIX ; xtx = <x’x> $ Computes an (X’X)^{-1} matrix.
```
7.5 The Current Sample of Observations

In many cases, you will simply analyze the entire data set that you input. But, most analyses also involve partitioning the data set into subsamples, either by stratification, or by excluding or including observations based on some data related criteria. This section will describe these two aspects of operation. The final section will include a discussion of how the sample is modified, either by you or automatically, when the data set contains missing observations.

**SAMPLE** designate specific observations to be included in a subsample,
**DATES** establish the periodicity of time series data,
**PERIOD** designate specific time series observations to be included in a subsample,
**REJECT** exclude certain observations from the sample based on an algebraic rule,
**INCLUDE** include certain observations from the sample based on an algebraic rule,

In most cases, you will read in a data set and use the full set of observations in your computations. But, it is quite common to partition the sample into subsamples and use its parts in estimation instead. You will also frequently want to partition the data set to define data matrices for use in the **MATRIX** commands.

NOTE: The ‘current sample’ is the set of observations, either part or all of an active data set, which is designated to be used in estimation and in the data matrices for **MATRIX**, **CREATE**, etc.

The commands described in this section are used to designate certain observations either ‘in’ or ‘out of’ the current sample. With only a few exceptions, operations which use your data, such as model estimation and data transformation, operate only on the current sample. For example, if you have initially read in 10 observations on \( x \) and \( y \), but then set the sample to include only observations 1-3, 6, and 8-10, nearly all commands will operate on or use only these seven observations. Thus, if you compute \( \log(x) \), only seven observations will be transformed.

To define the current sample, **LIMDEP** uses a set of switches, one for each observation in the data set. Thus, when you define the sample, you are merely setting these switches. As such, the **REJECT** command does not actually remove any data from the data set, it merely turns off some of these switches. The data are not lost. The observations are reinstated with a simple **SAMPLE** ; **ALL S**. Figure 7.15 shows the process. The sequence of instructions in the editing window creates a sample of draws from the standard normal distribution. The **SAMPLE** command chooses the first 12 of these observations, then the **REJECT** command removes from the sample observations that are greater than 1.0 or less than -1.0. This turns out to be observations 6 and 12, as can be seen in the data editor. The chevron to the right of the row number in the data editor is the switch discussed above.

There are two sets of commands for defining the current sample, one appropriate for cross section data and the other specifically for time series.
7.5.1 Cross Section Data

Initially, observations are defined with respect to ‘rows’ of the data matrix, which are simply numbered 1 to 1,000.

Sample Definition – The SAMPLE Command

Designate particular observations to be included in the current sample with the command

```
SAMPLE ; range, range, range, ..., range $
```

A ‘range’ is either a single observation number or a range of observations of the form lower-upper. For example,

```
SAMPLE ; 1, 12-35, 38, 44-301, 399 $
```

You can set the sample in this fashion, do the desired computations, then reset the sample to some other definition, at any time. To restore the sample to be the entire data set, use

```
SAMPLE ; All $
```

Because of the possibility of missing data being inadvertently added to your data set, *LIMDEP* handles this command as follows: ‘All’ observations are rows 1 to *N* where *N* is the last row in the data area which is not completely filled with missing data. In most cases, this will be the number of observations in the last data set you read. But, you can go beyond this last row by giving specific ranges on the command. For example, suppose you begin your session by reading a file of 100
observations. Thereafter, \texttt{SAMPLE ; All} would be equivalent to \texttt{SAMPLE ; 1-100}. But, you could then do the following:

\begin{verbatim}
SAMPLE ; 1 - 250
CREATE ; x = Rnn(0,1) (Random sample)
\end{verbatim}

Now, since there are 250 rows containing at least some valid data, \texttt{SAMPLE ; All} is equivalent to \texttt{SAMPLE ; 1-250}.

\section*{Exclusion and Inclusion – The REJECT and INCLUDE Commands}

These commands are used to delete observations from or add observations to the currently defined sample. They have the form

\begin{verbatim}
VERB ; logical expression
\end{verbatim}

‘VERB’ is either \texttt{REJECT} or \texttt{INCLUDE}. ‘Logical expression’ is any desired expression that provides the condition for the observation to be rejected or included. It may include any number of levels of parentheses and may involve mathematical expressions of any complexity involving variables, named scalars, matrix or vector elements, and literal numbers. The operators are as follows:

Math and relational operators are $+$, $-$, $*$, $/$, $^\wedge$, $>$, $\geq$, $<$, $\leq$, $=$, $\neq$.

Concatenation operators are $\&$ for ‘and’, $|$ for ‘or’.

A simple example appears in Figure 7.15. Another might be:

\begin{verbatim}
REJECT ; x > 0
\end{verbatim}

For a more complex example, we compute an expression for observations which are not inside a ball of unit radius.

\begin{verbatim}
REJECT ; x^2 + y^2 + z^2 >= 1
\end{verbatim}

The hierarchy of operations is $^\wedge$, $(\ast,/) (+,-)$, $(\geq,\leq,\leq,\neq)$, $\&$, $|$. Operators in parentheses have equal precedence and are evaluated from left to right. When in doubt, add parentheses. There is essentially no limit to the number of levels of parentheses. (They can be nested to about 20 levels.)

It is important to note that in evaluating expressions, you get a logical result, not a mathematical one. The result is either true or false. An expression which cannot be computed cannot be true, so it is false. Therefore, any subexpression which involves missing data or division by zero or a negative number to a noninteger power produces a result of false. But, that does not mean that the full expression is false. For example: \((x / 0) > 0 \mid x > y\) could be true. The first expression is false because of the zero divide, but the second might be true, and the ‘or’ in the middle returns ‘true’ if either expression is true. Also, we adopt the C language convention for evaluation of the truth of a mathematical expression. A nonzero result is true, a zero result is false. Thus, your expression need not actually make logical comparisons. For example: Suppose \(x\) is a binary variable (zeros and ones). \texttt{REJECT ; x} will reject observations for which \(x\) equals one, since the expression has a value of ‘true’ when \(x\) is not zero. Therefore, this is the same as \texttt{REJECT ; x \# 0}.
REJECT deletes observations from the currently defined sample while INCLUDE adds observations to the current sample. You can use either of these to define the current sample by writing your command as

REJECT or INCLUDE ; New ; ... expression ... $

For a REJECT command, this has the result of first setting the sample to All, then rejecting all observations which meet the condition specified in the expression. For an INCLUDE command, this has the effect of starting with no observations in the current sample and selecting for inclusion only those observations which meet the condition. In the latter case, this is equivalent to ‘selecting cases,’ as may be familiar to users of SAS or SPSS.

You may enter REJECT and INCLUDE commands from the dialog box shown in Figure 7.16. The dialog box is invoked by selecting Include or Reject in the Project:Set Sample menu or by right clicking in the data editor, clicking Set Sample, then selecting Reject or Include from the Set Sample menu. Note in the dialog box, the ‘Add observations to the current sample’ option at the top is the ; New specification in the command. Also, by clicking the query (?) button at the lower left, you can obtain information about these commands from the online Help file.

The same Set Sample menu offers All, which just generates a SAMPLE ; All $ command and Range which produces the dialog box shown in Figure 7.17.
TIP: If your REJECT command has the effect of removing all observations from the current sample, LIMDEP takes this as an error, gives you a warning that this is what you have done, and ignores the command.

Interaction of REJECT/INCLUDE and SAMPLE

REJECT and INCLUDE modify the currently defined sample unless you include ; New. But, SAMPLE always redefines the sample, in the process discarding all previous REJECT, INCLUDE, and SAMPLE commands. Thus,

SAMPLE ; 1-50,200-300 $

and

SAMPLE ; 1-50 $
SAMPLE ; 200-300 $

are not the same. The second SAMPLE command undoes then replaces the first one.

Any of these three commands may appear at any point, together or separately. Before any appear, the default sample is SAMPLE ; All $.

TIP: If you are using lagged variables, you should reset the sample to discard observations with missing data. This is generally not done automatically.

7.5.2 Time Series Data

When you are using time series data, it is more convenient to refer to rows of the data area and to observations by date, rather than by observation number. Two commands are provided for this purpose.

To give specific labels to the rows in the data area, use

DATES ; Initial date in sample $

The initial date may be one of:

Undated same as before. (Use this to undo a previous DATES command.)
DATES ; Undated $

YYYY year for yearly data, e.g., 1951.
DATES ; 1951 $

YYYY.Q year.quarter for quarterly data. Q must be 1, 2, 3, or 4.
DATES ; 1951.1 $

YYYY.MM year.month for monthly data. MM is 01 02 03 ... 12.
DATES ; 1951.04 $

Note that .1 is a quarter, and .5 is invalid. The fifth month is .05, and the tenth month is .10, not .1. Once the row labels are set up, the counterpart to the SAMPLE command is

PERIOD ; first period - last period $
For example,

```
PERIOD ; 1964.1 - 1977.4 S
```

These two commands do not change the way that any computations are done with LIMDEP. They will change the way certain output is labeled. For example, when you use the data editor, the row markers at the left will now be the dates instead of the observation numbers.

**NOTE:** You may not enter a date using only two digits. Your dates must contain all four digits. No computation that LIMDEP does or command that you submit that involves a date of any sort, for any purpose, uses two digits. Therefore, there is no circumstance under which LIMDEP could mistake 20xx for 19xx. Any two digit date submitted for any purpose will generate an error, and will not be processed.

The **DATES** command may be given from the **Project:Settings/Data Type** menu item.

![Figure 7.18 Dialog Box for DATES Command](image)

The **SAMPLE** and **PERIOD** commands may be given from the **Project:Set Sample Range** dialog box. See Figure 7.19.
NOTE: The current data type, Data:U, Data:Y, Data:Q, or Data:M is displayed at the top of the project window. The data editor will also be changed to show the time series data. Figure R7.7 shows an example using quarterly data. The top of the project window displays the ‘Q’ which indicates quarterly data. The data editor has also automatically adjusted following the setting in Figure 7.19 for quarterly data beginning in 1961.4.

7.6 Missing Data

Observations that contain missing values for variables in a model are automatically bypassed. (The full version of LIMDEP uses a SKIP command for this setting. The SKIP switch is automatically turned on in the Student version.) Missing values are handled specifically elsewhere in the program by the CREATE, CALCULATE and MATRIX commands.

The treatment of missing values by CALCULATE is as follows:

- Dot products involving variables: The procedure is aborted, and -999 is returned.
- Max and Min functions: Missing data are skipped.
- Lik, Rsq, etc. (regression functions): Same as dot products.

The matrix algebra program that directly accesses the data in several commands, including
x'x, for sums of squares and cross products, <x'x> for inverses of moment matrices, and many others will simply process them as if the -999s were legitimate values. Since it is not possible to deduce precisely the intention of the calculation, LIMDEP does not automatically skip these data or abort to warn you. It should be obvious from the results. You can specifically request this. If you do have the ‘SKIP’ switch set to ‘on’ during matrix computations (the student version does), LIMDEP will process MATRIX commands such as x'x and automatically skip over missing values. But, in such a case, the computation is usually erroneous, so your output will contain a warning that this has occurred, and you might want to examine closely the calculations being done to be sure it is really how you want to proceed.

Figure 7.20 illustrates the results discussed in the previous paragraph. The commands are shown in the editing window. Variables x and y are random samples of 100 observations from the standard normal distribution. The CREATE command changes a few observations in each column to missing values – the observations are not the same for x and y. The NAMELIST defines zx to be x and a column of ones – a two column matrix, and zy likewise. With SKIP turned on, the 2x2 matrix product zx'zy shows that there are 88 observations in the reduced sample (see the 88 that is 1'1 at the upper left corner) and a warning is issued. With SKIP turned off, in the second computation, the missing values are treated as -999s, and the resulting matrix has values that appear to be inappropriate.
This page intentionally left blank.
Chapter 8

Estimating Models

8.1 Introduction

Once your data are in place and you have set your desired current sample, most of your remaining commands will be either model estimation commands or the data manipulation commands, CREATE, CALCULATE, and MATRIX. We consider the model commands in this chapter and the other commands in Chapter 9. This chapter will describe the common form of all model estimation commands, estimation results, and how to produce useable output in an output file. Section 8.3 contains a general discussion on the important statistical features of the model estimators, such as using weights and interpreting results in effects. This chapter will also describe procedures that are generally used after a model is estimated, such as testing hypotheses, retrieving and manipulating results, and analyzing restrictions on model parameters. In terms of your use of LIMDEP for model estimation and analysis, this chapter is the most important general chapter in this part of the manual. The chapters in the second part of this manual will provide free standing and fairly complete descriptions of how to estimate specific models, but users are encouraged to examine Chapters 8 and 9 here closely for the essential background on these procedures.

8.2 Model Estimation Commands

Nearly all model commands are variants of the basic structure

MODEL COMMAND ; Lhs = dependent variable
; Rhs = list of independent variables
; ... other parts specific to the model ; ...

The 75 or so different models are specified by changing the model name or by adding or subtracting specifications from the template above. At different points, other specifications, such as Rh2 = a second list, are used to specify a list of variables. These will be described with the particular estimators. Different models will usually require different numbers and types of variables to be specified in the lists above. Note that, in general, you may always use namelists at any point where a list of variables is required. Also, a list of variables may be composed of a set of namelists.

| NOTE ON CONSTANT TERMS IN MODELS: Of the over 75 different models that LIMDEP estimates, only one, the linear regression model estimated by stepwise regression, automatically supplies a constant term in the Rhs list. If you want your model to contain a constant term, you must request it specifically by including the variable ‘one’ among your Rhs variables. You should notice this in all of our examples below. |

|
ADVICE ON MODEL SPECIFICATION: It is fairly rare that a model would be explicitly specified without a constant. In almost all cases, you should include the constant term. Omitting the constant amounts to imposing a restriction that will often distort the results, sometimes severely. (We recall a startling exchange among some of our users in reaction to what appeared to be drastic differences in the estimates of a probit model produced by LIMDEP and Stata. The difference turned out to be due to the omitted constant term in the LIMDEP command.) In a few cases, though it is not mandatory by the program, you definitely should consider the constant term essential – these would include the stochastic frontier and the ordered probit models. However, in a few other cases, you should not include a constant term. These are the fixed effects, panel data estimators, such as regression, logit, Poisson and so on. (In most cases, if you try to include an overall constant term in a fixed effects model, LIMDEP will remove it from the list.)

NOTE: With all of the different forms and permutations of features, LIMDEP supports several hundred different models. They are described fully in the 3 volume documentation for the full program. This student version contains all of those models, however, the manual will only describe a few of the available model specifications.

In addition to the specifications of variables, there are roughly 150 different specifications of the form

; sss [= additional information]

which are used to complete the model command. In some cases, these are mandatory, as in

REGRESS ; Lhs = ... ; Rhs = ... ; Panel ; Str = variable $.

This is the model command for the fixed and random effects linear regression model. The latter two specifications are necessary in order to request the panel data model. Without them, the command simply requests linear least squares. In other cases, specifications will be optional, as in

REGRESS ; Lhs = ... ; Rhs = ... ; Keep = yf $

which requests LIMDEP to fit a model by linear least squares, then compute a set of predictions and keep them as a new variable named yf.

Some model specifications are general and are used by most, if not all, of the estimation commands. For example, the ; Keep = name specification in the command above is used by all single equation models, linear or otherwise, to request LIMDEP to keep the predictions from the model just fit. In other cases, the specification may be very specific to one or only a few models. For example, the ; Cor in

SWITCHING REGRESSION ; Lhs = ... ; Rh1 = ... ; Rh2 = ... ; Cor $

is a special command used to request a particular variant of the switching regression model, that with correlation across the disturbances in the two regimes. The default is to omit ; Cor, which means no correlation.
8.2.1 The Command Builder

*LIMDEP* contains a set of dialog boxes and menus that you can use to build up your model commands in parts, as an alternative to laying out the model commands directly as shown above. Figure 8.1 shows the top level model selection. The menu items, *Data Description*, etc., are subsets of the modeling frameworks that *LIMDEP* supports. We’ve selected *Linear Models* from the menu, which produces a submenu offering *Regression*, *2SLS*, and so on. From here, the command builder contains specialized dialog boxes, specific to a particular model command.

![Figure 8.1 Model Command Builder with Linear Models Menu](image)

We’ll illustrate operation of the command builder with a familiar application, Grunfeld’s panel data set, 10 firms, 20 observations per firm, on three variables, investment, $i$, profit, $f$, and capital stock, $c$. The data are contained in the project shown in Figure 8.2. The variables, $d_1$, $d_2$, … are dummy variables for the 10 firms.

Figure 8.3 shows the main model specification dialog box (*Main* page), which will be quite similar for most of the models. The main window provides for specification of the dependent variable, the independent variables, and weights if desired. (Weights are discussed in Section 8.3.) We will not be using them in this example. If desired, the model is fully specified at this point. Note that the independent variables have been moved from window at the right, which is a menu, to the specification at the left. The highlighted variables D3 – D9 will be moved when we click the ‘<<’ button to select them. You may also click the query (?) button at the lower left of the dialog box to obtain a Help file description of the *REGRESS* command for linear models that is being assembled here. The Run button allows you now to submit the model command to the program to fit the model. There is also a box on the *Main* page for the *REGRESS* command for specifying the optional extension, the GARCH model. Since the main option box for this specification is not checked, this option will not be added to the model command.
Estimating Models

Figure 8.2 Project Window for the Grunfeld Data

Figure 8.3 Main Page of Command Builder for Linear Regression Model
The other two tabs in the command builder provide additional options for the linear regression model, as shown in Figures 8.4 and 8.5. For this example, we’ll submit the simple command from the Main page with none of the options. Clicking the Run button submits the command to the program, and produces the output shown in Figure R8.6.

The regression command that was assembled by the command builder can be seen in the output window directly above the regression output:

```
REGRESS ; Lhs = i ; Rhs = one,f,c,d1,d2,d3,d4 S
```
The command builder has done the work of constructing the command and sending it to the program. Although the command builders do not remember their previous commands, the commands are available for you to reuse if you wish. You can use edit copy/paste to copy commands from your output window into your editing window, then just submit them from the editing window. The advantage of this is that you now need not reenter the dialog box to reuse the command. For example, if you wanted to add a time trend, year, to this equation, you could just copy the command to the editor, add year to the Rhs list, then select the line and click GO.

**TIP:** Commands that are ‘echoed’ to the output window are always marked with the leading ‘-- >.’ The command reader will ignore these, so you can just copy and paste the whole line, or block of lines to move commands to your editing window.
The command builders are not complete for all models that can be specified by \textit{LIMDEP}. Many features, such as the newer panel data estimators, are not contained in the command builders. In fact, you will probably ‘graduate’ from the dialog boxes fairly quickly to using the text editor for commands. The editors provide a faster and more flexible means of entering program instructions.

\subsection*{8.2.2 Output from Estimation Programs}

Results produced by the estimation commands will, of course, vary from model to model. The display in the output window in Figure 8.7 would be typical. These are the results produced by estimation of a basic tobit model. (We converted the dependent variable in the preceding linear model to deviations from the overall mean then forced a tobit model on the resulting variable.)

Figure 8.7  Output Window for an Estimated Model
Displaying Covariance Matrices

One conspicuous absence from the output display is the estimate of the asymptotic covariance matrix of the estimates. Since models can have up to 150 parameters, this part of the output is potentially voluminous. Consequently, the default is to omit it. You can request that it be listed by adding

; Printvc

to the model command. Since covariance matrices can be extremely large, this is handled two ways. If the resultant matrix is quite small, it is included in the output. The earlier tobit equation is shown in Figure 8.10. If the matrix has more than five columns, then it is offered as an additional embedded matrix with the output, as shown in Figure 8.11 for a larger regression model. (We also included ; Matrix in this estimation. Note that there are two embedded matrices in the output.)

When estimation is done in stages ; Printvc will only produce an estimated covariance matrix at the final step. Thus, no covariance matrix is displayed for initial least squares results.

![Figure 8.10 Model Output with Covariance Matrix]
Figure 8.11 Regression Output with Embedded Covariance Matrix
8.3 Model Components and Results

The primary components of a model are provided with the model command

```
MODEL COMMAND ; Lhs = dependent variable
   ; Rhs = independent variables $
```

In many cases, this is all that is required. However there are a few common variations, notably using weights.

Constant Terms

As noted earlier, LIMDEP almost never forces a constant term in a model – the only case is the linear regression model fit by stepwise least squares. For any other case, if you want a constant term in a model, you must include the variable `one` in the appropriate variable list. The variable `one` is provided by the program; you do not have to create it. You can, however, use `one` at any point, in any model where you wish to have a constant term, and any `MATRIX` command based on a column of ones as a variable in the analysis of data. The typical estimation command will appear like the following example:

```
REGRESS ; Lhs = y ; Rhs = one, x $
```

which produces output including

```
+---------+--------------+----------------+--------+---------+----------+
|Variable | Coefficient  | Standard Error |b/St.Er.|P[Z>|z|] | Mean of X|
+---------+--------------+----------------+--------+---------+----------+
Constant -.00280548254    .0070621362      -.397   .6912
X         .01489733909    .0071169806      2.093   .0363    -.006473082
```

A `MATRIX` command based on the same construction might be the following which computes the inverse of a matrix of sums of squares and cross products:

```
NAMELIST  ; xdata = one,x $
MATRIX    ; xxi  = <x'x> $
```

8.3.1 Using Weights

Any procedure which uses sums of the data, including descriptive statistics and all regression and nonlinear models can use a weighting variable by specifying

```
; Wts = name
```

where `name` is the name of the variable to be used for the weighting.

Any model based on least squares of any sort or on likelihood methods can be estimated with weights. This includes `REGRESS`, `PROBIT`, all `LOGIT` models, and so on. The only substantive exceptions are the nonparametric and semiparametric estimators, `MSCORE`, `NPREG`, and the Cox proportional hazard model.
NOTE: In computing weighted sums, the value of the variable, not its square root is used. As such, if you are using this option to compute **weighted least squares for a heteroscedastic regression**, name should contain the reciprocals of the disturbance variances, not the standard deviations.

In maximum likelihood estimation, the terms in the log likelihood and its derivatives, not the data themselves, are multiplied by the weighting variable. That is, when you provide a weighting variable, **LIMDEP** computes a sum of squares and cross products in a matrix as $X'WX = \Sigma w_i x_i x_i'$ and a log likelihood $\log L = \Sigma w_i \log(f_i)$, where $w_i$ is an observation on your weighting variable.

The weighting variable must always be positive. The variable is examined before the estimation is attempted. If any nonpositive values are found, the estimation is aborted.

During computation, weights are automatically scaled so that they sum to the current sample size. The variable, itself, is not changed, however. If you specify that variable $w$ is to be the weighting variable in ; Wts = w, the weight actually applied is $w_i^* = [N/\Sigma w_i] \times w_i$. This scaling may or may not be right for a selected sample in a sample selection model. That is, after selection, the weights on the selected data points may or may not sum to the number of selected data points. As such, the weights in **SELECT** with univariate and bivariate probit criterion equations are rescaled so that they sum exactly to the number of selected observations.

**TIP:** The scaling will generally not affect coefficient estimates. But, it will affect estimated standard errors, sometimes drastically.

To suppress the scaling, for example for a grouped data set in which the weight is a replication factor, use

; Wts = name,noscale or just ; Wts = name,n

**WARNING:** When this option is used with grouped data qualitative choice models, such as logit, it often has the effect of enormously reducing standard errors and blowing up t-ratios.

The ‘noscale’ option would most likely be useful when examining proportions data with a known group size. For example, consider a probit analysis of county voting returns. The data would consist of $N$ observations on $[n_i, p_i, x_i]$, where $n_i$ is the county size, $p_i$ is the proportion of the county population voting on the issue under study, and $x_i$ is the vector of covariates. Such data are heteroscedastic, with the variance of the measured proportion being proportional to $1/n_i$. We emphasize, once again, when using this option with population data, standard errors tend to become vanishingly small, and call upon the analyst to add the additional measure of interpretation.

### 8.3.2 Model Output

Most of the models estimated by **LIMDEP** are single equation, ‘index function’ models. That is, there is a dependent variable, which we’ll denote ‘$y$,’ a set of independent variables, ‘$x$,’ and a model, consisting, in most cases, of either some sort of regression equation or a statement of a probability distribution, either of which depends on an index function, $x' \beta$ and a set of ‘ancillary’ parameters, $\theta$, such as a variance term, $\sigma^2$ in a regression or a tobit model. The parameters to be estimated are $[\beta, \theta]$. In this framework, estimation will usually begin with a least squares regression of $y$ on $x$. This will often be for the purpose of obtaining the default starting values for the iterations, but is sometimes done simply because in this modeling framework, the ordinary least squares (OLS) estimate is often an interesting entity in its own right. Model output, not including the technical output from the iterations (see Section R8.4) will thus consist of the OLS results, followed by the primary objects of estimation.
NOTE: In order to reduce the amount of superfluous output, OLS results are not reported automatically except for the linear regression model. To see the OLS outputs when they are computed, add ; OLS to your model command.

TIP: It is important to keep in mind that the OLS estimates are almost never consistent estimates of the parameters of the nonlinear models estimated by LIMDEP.

Listed below are the results from estimation of a Poisson regression model. (The example is used at various points in the chapters to follow.) The model command requests a set of marginal effects. These are discussed in the next section.

The following results are reported when your model command contains ; OLS:

```
+----------------------------------------------------+
| Poisson Regression Model - OLS Results             |
| Ordinary least squares regression                  |
| LHS=NUM Mean = 10.47059                             |
| WTS=none Number of observs. = 34                    |
| Model size Parameters = 9                          |
| Degrees of freedom = 25                            |
| Sum of squares = 2076.755                           |
| Standard error of e = 9.114286                      |
| R-squared = .7458219                                |
+----------------------------------------------------+
```

| Variable | Coefficient | Standard Error |b/St.Er.|P[|Z|>z] | Mean of X|
|----------|-------------|----------------|--------|--------|----------|
| Constant | -39.6343740 | 9.30415297     -4.260 | .0000  |
| A        | 4.38284585  | 5.12356623     .855  | .3923  | .23529412 |
| C        | -1.60558349 | 4.85887959     -.350 | .7411  | .23529412 |
| D        | 1.75161158  | 4.99174267     .351  | .7257  | .23529412 |
| E        | -7.29195392 | 5.63651443     -1.294 | .1958  | .14705882 |
| LOGMTH   | 7.30381292  | 1.36910416     5.335  | .0000  | 7.04925451 |
```
The following are the standard results for a model estimated by maximum likelihood or some other technique. Most parts are common, while a few are specific to the particular model.

<table>
<thead>
<tr>
<th>Poisson Regression</th>
<th>Maximum Likelihood Estimates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dependent variable NUM</td>
<td>Number of observations 34</td>
</tr>
<tr>
<td>Log likelihood function -72.69771</td>
<td>Number of parameters 9</td>
</tr>
<tr>
<td>Info. Criterion: AIC = 4.80575</td>
<td>Finite Sample: AIC = 5.02634</td>
</tr>
<tr>
<td>Restricted log likelihood -356.2029</td>
<td>Chi squared 567.0104</td>
</tr>
<tr>
<td>Degrees of freedom 8</td>
<td>Degrees of freedom 8</td>
</tr>
<tr>
<td>Prob[ChiSq &gt; value] = .0000000</td>
<td></td>
</tr>
<tr>
<td>Chi-squared = 46.55065 RsqP = .9403</td>
<td></td>
</tr>
<tr>
<td>G-squared = 47.52893 RsqD = .9227</td>
<td></td>
</tr>
<tr>
<td>Overdispersion tests: g=mu(i) : 1.770</td>
<td></td>
</tr>
<tr>
<td>Overdispersion tests: g=mu(i)^2 : .498</td>
<td></td>
</tr>
</tbody>
</table>

| Variable | Coefficient | Standard Error | b/St.Er. | P[|Z|>z] | Mean of X |
|---------|--------------|----------------|---------|---------|----------|
| Constant | -4.80807969 | .65751784 | -7.312 | .0000 | |
| A       | .00373503   | .15767690 | .024   | .9811  | .23529412 |
| C       | -.83536379  | .36686744 | -2.277 | .0228  | .23529412 |
| D       | .29514731   | .24430603 | 1.208  | .2270  | .23529412 |
| E       | .13092973   | .32043868 | .409   | .6828  | .05882353 |
| C67     | .70947001   | .17792942 | 3.987  | .0001  | .29411765 |
| C72     | .86911092   | .24440891 | 2.101  | .0357  | .14705882 |
| C77     | .51343007   | .06534072 | 12.289 | .0000  | 7.04925451 |

Some notes about the results shown above:

- The sample was changed from the initial 40 observations to the 34 observations with a nonmissing value of num before estimation.
- The least squares results are the standard set of results that will appear with every ordinary least squares regression. However, the table of diagnostic statistics shown above the coefficient estimates is slightly abridged when the model command is not specifically for a linear regression. The linear regression model would show autocorrelation diagnostics and other information criteria.
- Results always include the coefficients, standard errors, and ratios of coefficients to standard errors. In the index function models, the coefficients are named by the variable that multiplies them in the index function. In models which do not use an index function (NLSQ, MINIMIZE, CLOGIT, and a few others), the parameter label that you provide will appear with the estimate instead.
- Results for index function models include the mean values of the variables that are multiplied by the coefficients. Ancillary parameters and user defined parameters will show blanks in this part of the table. Note that ‘one’ becomes ‘Constant’ in the table.
- The prob value shown, ‘P[|Z| > z],’ is the value for a two tailed test of the hypothesis that
the coefficient equals zero. The probability shown is based on the standard normal distribution in all cases except the linear regression model, when it is based on the ‘t’ distribution with degrees of freedom that will be shown in the table.

- The diagnostics table for the Poisson regression reports some statistics which will be present for all models (indicated by the box in the figure):
  1. left hand side variable,
  2. number of observations used,
  3. number of iterations completed,
  4. log likelihood function or other estimation criterion function.

- Some results will be computable only for some models. The following results listed for the Poisson model will not appear when there is no natural, nested hypothesis to test. (For example, they will not normally appear in the output for the tobit model.)
  1. log likelihood at a restricted parameter estimate, usually zero,
  2. chi squared test of the restriction,
  3. significance level,
  4. degrees of freedom.

- Finally, there are usually some statistics or descriptors which apply specifically to the model being estimated. In this example, chi squared, \( G^2 \) squared and two overdispersion tests are statistics that are specific to the Poisson model.

### 8.3.3 Retrievable Results

When you estimate a model, the estimation results are displayed on the screen and in the output file if one is open. In addition, each model produces a number of results which are saved automatically and can be used in subsequent procedures and commands. The **POISSON** command above shows an example. After the model is estimated, scalars named \( nreg, kreg, \) and \( \logl \) are created and set equal to the number of observations, number of coefficients estimated, and the log likelihood for the model, respectively. For another, after you give a **REGRESS** command, the scalar \( rsqr \) is thereafter equal to the \( R^2 \) from that regression. You can retrieve these and use them in later commands. For example,

```
REGRESS ; ...$
CALC ; f = rsqr/(kreg-1) / ((1 - rsqr)/(nreg - kreg)) $
```

to compute a standard \( F \) statistic. (There is an easier way to do this.)

Although your **CALCULATOR** has 50 cells, the first 14 are ‘read only’ in the sense that **LIMDEP** reserves them for estimation results. You may use these scalars in your calculations, or in other commands (see the example above), but you may not change them. (The one named \( \rho \) may be changed.) Likewise, the first three matrices are reserved by the program for ‘read only’ purposes. The read only scalars are

```
ssqr, rsqr, s, sumsqdef, degfrdm, ybar, sy, kreg, nreg, \logl, exitcode
```

and two whose names and contents will depend on the model just estimated. The names used for these will be given with the specific model descriptions. At any time, the names of the read only scalars are marked in the project window with the \( \mathbb{1} \) symbol to indicate that these names are ‘locked.’ Figure 8.12 illustrates. This shows the setup of the project window after the tobit example developed above.
A parameter vector is automatically retained in a matrix named $b$. The program will also save the estimated asymptotic covariance matrix and name it $varb$. The reserved matrices are thus $b$ and $varb$, with a third occasionally used and renamed. The third, protected matrix name will depend on the model estimated. A few examples are:

- $mu$ created by ORDERED PROBIT,
- $sigma$ created by SURE and 3SLS,
- $pacf$ created by IDENTIFY.

All estimators set at least some of these matrices and scalars. In the case of the scalars, those not saved by the estimator are set to 0. For example, the PROBIT estimator does not save $rsqrd$. Matrices are simply left unchanged. So, for example, if you estimate a fixed effects model, which creates the third matrix and calls it $alphafe$, then estimate a probit model which only computes $b$ and $varb$, $alphafe$ will still be defined.
TIP: Each time you estimate a model, the contents of \( b \), \( \text{varb} \), and the scalars are replaced. If you do not want to lose the results, retain them by copying them into a different matrix or scalar. For example, the following computes a Wald test statistic for the hypothesis that the slope vector in a regression is the same for two groups (a Chow test of sorts):

```plaintext
REGRESS ; Lhs = y ; Rhs = ... ; If [Male = 1] $
MATRX ; b\text{male} = b ; v\text{male} = \text{varb}$
REGRESS ; Lhs = y ; Rhs = ... ; If [Female = 1] $
MATRX ; b\text{female} = b ; v\text{female} = \text{varb}$
MATRX ; d = b\text{male} - b\text{female}
; waldstat = d' * \text{Nvsm}(v\text{male}, v\text{female}) * d$
```

The matrix results saved automatically in \( b \) and \( \text{varb} \) are, typically, a slope vector, \( b \), and the estimated asymptotic covariance matrix of the estimator, from an index function model. For example, when you estimate a tobit model, the estimates and asymptotic covariance matrix are

\[
\begin{bmatrix}
\beta \\
\sigma
\end{bmatrix} \quad \text{and} \quad
\begin{bmatrix}
V_{\beta\beta} & V_{\beta\sigma} \\
V_{\sigma\beta} & V_{\sigma\sigma}
\end{bmatrix}
\]

The results kept are \( \beta \), in \( b \), \( V_{\beta\beta} \) in \( \text{varb} \), and \( \sigma \) in a scalar named \( s \). The other parts of the asymptotic covariance matrix are generally discarded. We call the additional parameters, such as \( s \), the ancillary parameters in the model. Most of the models that LIMDEP estimates contain one or two ancillary parameters. These are generally handled as in this example; the slope vector is retained as \( b \), the ancillary parameters are kept as named scalars, and the parts of the covariance matrix that apply to them are discarded.

In some applications, you may want the full parameter vector and covariance matrix. You can retain these, instead of just the submatrices listed above, by adding the specification

; Parameters

(or, just ; Par) to your model command. (Note, for example, the computation of marginal effects for a dummy variable in a tobit model developed in the previous section.) Without this specification, the saved results are exactly as described above. The specific parameters saved by each command are listed with the model application in the chapters to follow. You will find an example of the use of this parameter setting in the program for marginal effects for a binary variable in the tobit model, which is in the previous section.

### 8.3.4 Creating and Displaying Predictions and Residuals

Most of the single equation models in LIMDEP, though not all, contain a natural ‘dependent variable.’ Model predictions for any such model are easily obtained as discussed below. What constitutes a residual in these settings is a bit ambiguous, but, once again, some construction that typically reflects a deviation of an actual from a predicted value can usually be retained. The exact definition of a ‘fitted value’ and a ‘residual’ are given with the model descriptions in the chapters to follow.

There are several options for computing and saving fitted values from the regression models. You may request fitted values and/or residuals for almost any model. (The exceptions are, e.g., multiple equation models.) The fitted values are requested by adding
; Keep = name
to your model command. The request for residuals is

; Res = name

In each of these cases, the command will overwrite the variable if it already exists, or create a new
one. In any model command, the specification

; List

requests a listing of the residuals and several other variables.

**TIP:** To keep fitted values in a text file, you can either use ; List with an output file or use WRITE
and write the values in their own file or LIST ; variable $.

If the current sample is not the entire data set, and the data array contains observations on
the regressors but not the dependent variable, you can interpolate and produce predicted values for
these observations by adding the specification

; Fill

to your model command. ; Res, ; Keep, and ; Fill do not compute values for any observations for
which any variable to be used in the calculation is missing (i.e., equals -999). Otherwise, a
prediction is computed for every row for which data can be found.

**TIP:** ; Fill provides a very simple way of generating out of sample predictions.

To provide an example of the ; Fill feature, we will examine some data on gasoline sales in
the U.S. before and after the 1973-1974 oil embargo. The data below are yearly series on gasoline
sales ($g$), per capita income ($y$), and index numbers for a number of prices: $pg$ is the gasoline price,
$pnc$, $puc$, and $ppt$ are price indices for new and used cars and public transportation, and $pn$, $pd$, and
$ps$ are aggregate price indices for nondurables, durables, and services.

```
READ ; Nobs = 27 ; Nvar = 10 ; Names = year,g,pg,y,pnc,puc,ppt,pd,pn,ps $
1960 129.7 .925 6036 1.045 .836 .810 .444 .331 .302
1961 131.3 .914 6113 1.045 .869 .846 .448 .335 .307
1962 137.1 .919 6271 1.041 .948 .874 .457 .338 .314
1963 141.6 .918 6378 1.035 .960 .885 .463 .343 .320
1964 148.8 .914 6727 1.032 1.001 .901 .470 .347 .325
1965 155.9 .949 7027 1.009 .994 .919 .471 .353 .332
1966 164.9 .970 7280 .991 .970 .952 .475 .366 .342
1967 171.0 1.000 7513 1.000 1.000 1.000 .483 .375 .353
1968 183.4 1.014 7728 1.028 1.028 1.046 .501 .390 .368
1969 195.8 1.047 7891 1.044 1.031 1.127 .514 .409 .386
1970 207.4 1.056 8134 1.076 1.043 1.285 .527 .427 .407
1971 218.3 1.063 8322 1.120 1.102 1.377 .547 .442 .431
1972 226.8 1.076 8562 1.110 1.105 1.434 .555 .458 .451
1973 237.9 1.181 9042 1.111 1.176 1.448 .566 .497 .474
1974 225.8 1.599 8867 1.175 1.226 1.480 .604 .572 .513
1975 232.4 1.708 8944 1.276 1.464 1.586 .659 .615 .556
1976 241.7 1.779 9175 1.357 1.679 1.742 .695 .638 .598
1977 249.2 1.882 9381 1.429 1.828 1.824 .727 .671 .648
1978 261.3 1.963 9735 1.538 1.865 1.878 .769 .719 .698
1979 248.9 2.656 9829 1.660 2.010 2.003 .821 .800 .756
1980 226.8 3.691 9722 1.793 2.081 2.516 .892 .894 .839
```
We will compute simple regressions of $g$ on $one$, $pg$, and $y$. The first regression is based on the pre-embargo data, 1960-1973, but fitted values are produced for all 27 years. The second regression uses the full data set and also produces predicted values for the full sample. We then plot the actual and both predicted series on the same figure to examine the influence of the later data points.

<table>
<thead>
<tr>
<th>Year</th>
<th>$g$</th>
<th>$pg$</th>
<th>$y$</th>
<th>$\hat{g}$</th>
<th>$\hat{pg}$</th>
<th>$\hat{y}$</th>
<th>$\hat{g}_1$</th>
<th>$\hat{g}_2$</th>
<th>$\hat{g}_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1981</td>
<td>225.6</td>
<td>4.109</td>
<td>9769</td>
<td>1.902</td>
<td>2.569</td>
<td>3.120</td>
<td>.957</td>
<td>.969</td>
<td>.926</td>
</tr>
<tr>
<td>1982</td>
<td>228.8</td>
<td>3.894</td>
<td>9725</td>
<td>1.976</td>
<td>2.964</td>
<td>3.460</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
</tr>
<tr>
<td>1983</td>
<td>239.6</td>
<td>3.764</td>
<td>9930</td>
<td>2.026</td>
<td>3.297</td>
<td>3.626</td>
<td>1.041</td>
<td>1.021</td>
<td>1.062</td>
</tr>
<tr>
<td>1984</td>
<td>244.7</td>
<td>3.707</td>
<td>10421</td>
<td>2.085</td>
<td>3.757</td>
<td>3.852</td>
<td>1.038</td>
<td>1.050</td>
<td>1.117</td>
</tr>
<tr>
<td>1985</td>
<td>245.8</td>
<td>3.738</td>
<td>10563</td>
<td>2.152</td>
<td>3.797</td>
<td>4.028</td>
<td>1.045</td>
<td>1.075</td>
<td>1.173</td>
</tr>
<tr>
<td>1986</td>
<td>269.4</td>
<td>2.921</td>
<td>10780</td>
<td>2.240</td>
<td>3.632</td>
<td>4.264</td>
<td>1.053</td>
<td>1.069</td>
<td>1.224</td>
</tr>
</tbody>
</table>

The following chapters will detail the formulas used in computing predictions, residuals, and accompanying information. When you use ; List, some additional information will be displayed in your output. In some cases, there is no natural residual or prediction to be computed, for example in the bivariate probit model. In these cases, an alternative computation is done, so what is requested by ; Res or ; Keep may not actually be a residual or a fitted value. Individual model
Estimating Models 8-19

Descriptions will provide details. In general, the ; List specification produces the following:

1. an indicator of whether the observation was used in estimating the model. If not, the observation is marked with an asterisk,
2. the observation number or date if the data are time series,
3. the observed dependent variable when this is well defined,
4. the ‘fitted value’ = variable retained by ; Keep,
5. the ‘residual’ = variable retained by ; Res,
6. ‘variable 1,’ a useful additional function of the model which is not kept, and
7. ‘variable 2,’ another computation.

Although the last two variables are not kept internally, they are written to your output window and to the output file if one is open, so you can retrieve them later by editing the file with a word processor. In all cases, the formulas for these variables will be given, so if you need to have them at the time they are computed, you can use a subsequent CREATE command to obtain the variables.

We illustrate these computations with a Poisson regression and with the out of sample predictions generated by the regression above. The POISSON command would be

```
POISSON ; Lhs = … ; Rhs = … ; List $
```

The following table results and items listed for the Poisson model are:

- Actual: $y$, Prediction: $E[y] = \exp(b'x)$,
- Residual: $y - E[y]$, Index: $b'x$,
- Probability: $Pr[Y = y] = \exp(-\lambda)\lambda^y/y!$, $\lambda = E[y]$.

LIMDEP Estimation Results Run log line 6 Page 3
Current sample contains 34 observations.

<table>
<thead>
<tr>
<th>Observation</th>
<th>Observed Y</th>
<th>Predicted Y</th>
<th>Residual</th>
<th>95% Forecast Interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0000</td>
<td>0.39914</td>
<td>-0.3991</td>
<td>-0.9184</td>
</tr>
<tr>
<td>2</td>
<td>0.0000</td>
<td>0.22733</td>
<td>-0.2273</td>
<td>-1.4813</td>
</tr>
<tr>
<td>3</td>
<td>3.0000</td>
<td>4.5761</td>
<td>-1.5761</td>
<td>1.5209</td>
</tr>
<tr>
<td>4</td>
<td>4.0000</td>
<td>4.5761</td>
<td>-0.5761</td>
<td>1.5209</td>
</tr>
<tr>
<td>5</td>
<td>6.0000</td>
<td>6.9559</td>
<td>-0.9559</td>
<td>1.9396</td>
</tr>
<tr>
<td>6</td>
<td>18.000</td>
<td>13.185</td>
<td>4.8150</td>
<td>2.5791</td>
</tr>
<tr>
<td>8</td>
<td>11.000</td>
<td>6.6922</td>
<td>4.3078</td>
<td>1.9009</td>
</tr>
<tr>
<td>9</td>
<td>39.000</td>
<td>44.388</td>
<td>-5.3881</td>
<td>3.7930</td>
</tr>
<tr>
<td>10</td>
<td>29.000</td>
<td>20.603</td>
<td>8.3967</td>
<td>3.0255</td>
</tr>
</tbody>
</table>

For a linear regression, the listed items are the familiar ones:

```
REGRESS ; Lhs = g ; Rhs = one, pg, y ; Keep = gfit6073 ; List ; Fill $
```

The following table results and items listed for the regression model are:

```
Predicted Values (* => observation was not in estimating sample.)
<table>
<thead>
<tr>
<th>Observation</th>
<th>Observed Y</th>
<th>Predicted Y</th>
<th>Residual</th>
<th>95% Forecast Interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>1960</td>
<td>129.70</td>
<td>127.98</td>
<td>1.7243</td>
<td>113.5586 142.3929</td>
</tr>
<tr>
<td>1961</td>
<td>131.30</td>
<td>129.47</td>
<td>1.8315</td>
<td>115.8623 143.0746</td>
</tr>
<tr>
<td>1962</td>
<td>137.10</td>
<td>134.80</td>
<td>2.3032</td>
<td>121.5356 148.0580</td>
</tr>
<tr>
<td>1963</td>
<td>141.60</td>
<td>138.04</td>
<td>3.5551</td>
<td>124.9718 151.1179</td>
</tr>
<tr>
<td>1964</td>
<td>148.80</td>
<td>148.58</td>
<td>0.2217</td>
<td>134.9251 162.2315</td>
</tr>
<tr>
<td>1965</td>
<td>155.90</td>
<td>160.79</td>
<td>-4.8910</td>
<td>147.7508 173.8312</td>
</tr>
<tr>
<td>1966</td>
<td>164.90</td>
<td>170.39</td>
<td>-5.4906</td>
<td>157.4244 183.3568</td>
</tr>
<tr>
<td>1967</td>
<td>171.00</td>
<td>180.11</td>
<td>-9.1072</td>
<td>167.4143 192.8002</td>
</tr>
<tr>
<td>1968</td>
<td>183.40</td>
<td>187.95</td>
<td>-4.5490</td>
<td>175.1447 200.7533</td>
</tr>
<tr>
<td>1969</td>
<td>195.80</td>
<td>195.73</td>
<td>0.0665</td>
<td>182.9210 208.5460</td>
</tr>
<tr>
<td>1970</td>
<td>207.40</td>
<td>204.04</td>
<td>3.3641</td>
<td>191.1278 216.9441</td>
</tr>
<tr>
<td>Year</td>
<td>Value1</td>
<td>Value2</td>
<td>Value3</td>
<td>Value4</td>
</tr>
<tr>
<td>------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>1971</td>
<td>218.30</td>
<td>210.46</td>
<td>7.8377</td>
<td>197.3139</td>
</tr>
<tr>
<td>1972</td>
<td>226.80</td>
<td>219.00</td>
<td>7.8000</td>
<td>205.4731</td>
</tr>
<tr>
<td>1973</td>
<td>237.90</td>
<td>242.57</td>
<td>-4.6662</td>
<td>226.7579</td>
</tr>
<tr>
<td>*1974</td>
<td>225.80</td>
<td>271.46</td>
<td>-45.6629</td>
<td>197.8524</td>
</tr>
<tr>
<td>*1975</td>
<td>232.40</td>
<td>282.81</td>
<td>-50.4150</td>
<td>193.9795</td>
</tr>
<tr>
<td>*1976</td>
<td>241.70</td>
<td>295.84</td>
<td>-54.1380</td>
<td>199.1641</td>
</tr>
<tr>
<td>*1977</td>
<td>249.20</td>
<td>310.71</td>
<td>-61.5120</td>
<td>201.1378</td>
</tr>
<tr>
<td>*1978</td>
<td>261.30</td>
<td>328.38</td>
<td>-67.0848</td>
<td>201.1378</td>
</tr>
<tr>
<td>*1979</td>
<td>248.90</td>
<td>388.25</td>
<td>-139.3480</td>
<td>168.3394</td>
</tr>
<tr>
<td>*1980</td>
<td>226.80</td>
<td>469.95</td>
<td>-243.1545</td>
<td>93.6745</td>
</tr>
<tr>
<td>*1981</td>
<td>225.60</td>
<td>505.76</td>
<td>-280.1606</td>
<td>67.3596</td>
</tr>
<tr>
<td>*1982</td>
<td>228.80</td>
<td>486.73</td>
<td>-257.9265</td>
<td>80.0424</td>
</tr>
<tr>
<td>*1983</td>
<td>239.60</td>
<td>482.43</td>
<td>-242.8259</td>
<td>97.6831</td>
</tr>
<tr>
<td>*1984</td>
<td>244.70</td>
<td>493.02</td>
<td>-248.3244</td>
<td>122.6819</td>
</tr>
<tr>
<td>*1985</td>
<td>245.80</td>
<td>499.99</td>
<td>-254.1909</td>
<td>126.6925</td>
</tr>
<tr>
<td>*1986</td>
<td>269.40</td>
<td>439.62</td>
<td>-170.2191</td>
<td>191.3035</td>
</tr>
</tbody>
</table>
Chapter 9

Using Matrix Algebra

9.1 Introduction

The data manipulation and estimation programs described in the chapters to follow are part of LIMDEP’s general package for data analysis. The MATRIX, CREATE, and CALCULATE commands provide most of the additional tools. By defining data matrices with the NAMELIST, SAMPLE, REJECT, INCLUDE, PERIOD, and DRAW commands, you can arbitrarily define as many data matrices as you want. Simple, compact procedures using MATRIX commands will then allow you to obtain covariance and correlation matrices, condition numbers, and so on. More involved procedures can be used in conjunction with the other commands to program new, possibly iterative, estimators, or to obtain complicated marginal effects or covariance matrices for two step estimators.

To introduce this extensive set of tools and to illustrate its flexibility, we will present two examples. (These are both built in procedures in LIMDEP, so the matrix programs are only illustrative.) The rest of the chapter will provide some technical results on matrix algebra and material on how to use MATRIX to manipulate matrices.

Example Restricted Least Squares

In the linear regression model, \( y = X\beta + \epsilon \), the linear least squares coefficient vector, \( \mathbf{b}^* \), and its asymptotic covariance matrix, computed subject to the set of linear restrictions \( \mathbf{Rb}^* = \mathbf{q} \) are

\[
\mathbf{b}^* = \mathbf{b} - (\mathbf{X}'\mathbf{X})^{-1}\mathbf{R}'[(\mathbf{R}\mathbf{X}'\mathbf{X})^{-1}\mathbf{R}']^{-1}(\mathbf{Rb} - \mathbf{q}),
\]

where \( \mathbf{b} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y} \)

and

\[
\text{Est.Asy.Var}[\mathbf{b}^*] = s^2(\mathbf{X}'\mathbf{X})^{-1} - s^2(\mathbf{X}'\mathbf{X})^{-1}\mathbf{R}'[(\mathbf{R}\mathbf{X}'\mathbf{X})^{-1}\mathbf{R}']^{-1}\mathbf{R}(\mathbf{X}'\mathbf{X})^{-1}.
\]

First, define the \( \mathbf{X} \) matrix, columns then rows. We assume the dependent variable is \( y \).

```
NAMELIST ; x = ... $  
CREATE ; y = the dependent variable $  
SAMPLE ; ... as appropriate $  
```

Next, define \( \mathbf{R} \) and \( \mathbf{q} \). This varies by the application. Get the inverse of \( \mathbf{X}'\mathbf{X} \) now, for convenience.

```
MATRIX ; r = ... ; q = ... ; xxi = <x'x> $  
```

Compute the unrestricted least squares and the discrepancy vector.

```
MATRIX ; bu = xxi * x'y ; d = r * bu - q $  
```
Compute the restricted least squares estimates and the sum of squared deviations.

\[
\text{MATRIX} \quad \text{br} = \text{bu} - \text{xxi}^* \text{r}^* \text{Iprd(r,xxi,r')} * \text{d} $
\]

\[
\text{CREATE} \quad \text{u} = \text{y} - \text{x}'\text{br} $
\]

Compute the disturbance variance estimator.

\[
\text{CALC} \quad \text{s2} = (1/(\text{Col(x)}+\text{Row(r)})) * \text{u}'\text{u} $
\]

Compute the covariance matrix, then display the results.

\[
\text{MATRIX} \quad \text{vr} = \text{s2} * \text{xxi} - \text{s2} * \text{xxi}^* \text{r}^* \text{Iprd(r,xxi,r')} * \text{r} * \text{xxi} \\
; \text{Stat(br,vr,x)} $
\]

The preceding gives the textbook case for obtaining the restricted least squares coefficient vector when \(X'X\) is nonsingular. For the case in which there is multicollinearity, but the restrictions bring the problem up to full rank, the preceding is inadequate. (See Greene and Seaks (1991).) The general solution to the restricted least squares problem is provided by the partitioned matrix equation:

\[
\begin{bmatrix}
X'X & R' \\
R & 0
\end{bmatrix}
\begin{bmatrix}
b^* \\
\lambda
\end{bmatrix}
= 
\begin{bmatrix}
X'y \\
q
\end{bmatrix}
\]

If the matrix in brackets can be inverted, then the restricted least squares solution is obtained along with the vector of Lagrange multipliers, \(\lambda\). The estimated asymptotic covariance matrix will be the estimate of \(\sigma^2\) times the upper left block of the inverse. If \(XX\) has full rank, this coincides with the solution above. The routine for this computation is

\[
\text{MATRIX} \quad \text{xx} = \text{x}'\text{x} \quad \text{xy} = \text{x}'\text{y} \quad \text{r} = \ldots \quad \text{q} = \ldots $
\]

\[
\text{CALC} \quad \text{k} = \text{Col(x)} \quad \text{j} = \text{Row(r)} $
\]

\[
\text{MATRIX} \quad \text{zero} = \text{Init}(\text{j}, \text{j}, 0) \\
; \text{a} = [\text{xx} / \text{r}, \text{zero}] \quad \text{Shorthand for symmetric partitioned matrix} \\
; \text{v} = [\text{xy} / \text{q}] \\
; \text{ai} = \text{Ginv(a)}; \text{b}_l = \text{ai} * \text{v} \\
; \text{br} = \text{b}_l(1:\text{k}) \quad \text{vr} = \text{ai}(1:\text{k}, 1:\text{k}) $
\]

\[
\text{CREATE} \quad \text{u} = \text{y} - \text{x}'\text{br} $
\]

\[
\text{MATRIX} \quad \text{vr} = \{\text{u}'\text{u} / (\text{n}-\text{k}+\text{j})\} * \text{vr} \quad \text{Stat(br,vr,x)} $
\]

### 9.2 Entering MATRIX Commands

MATRIX commands are typically given as parts of programs that perform larger functions, such as in the examples in Section 9.2.1. You also have a matrix ‘calculator’ that you can access occasionally in a window that is separate from your primary desktop windows (project, editing, and output).

#### 9.2.1 The Matrix Calculator

You may invoke the matrix calculator by selecting Tools:Matrix Calculator as shown in Figure 9.1.
The matrix calculator window is shown in Figure 9.2.

You can leave the matrix calculator window open while you go to some other function. For example, you may find it convenient to interrupt your work in the editing/output windows by activating the matrix calculator to check some result which, perhaps, is not emerging the way you expected.

There are two other ways to enter commands in the matrix calculator window. You can type \texttt{MATRIX} commands in the smaller ‘Expr:’ (expression) window. In this dialog mode, if your command will not fit on one line, just keep typing. At some convenient point, the cursor will automatically drop down to the next line. Only press Enter when you are done entering the entire command. In this mode of entry, you do not have to end your commands with a $$. 
Alternatively, you can click the $f_x$ button to open a subsidiary window that provides a menu of the functions (procedures): See Figure 9.3.

![Insert Function Window](image)

Figure 9.3 Insertion Window for Matrix Functions

You can select the function you wish to insert in your command. You must then fill in the arguments of the function that are specific to your expression. (E.g., if you want \texttt{Chol(sigma)}, you can select \texttt{Chol(A)} from the menu, then you must change ‘A’ to ‘sigma.’) in the command.

### 9.2.2 MATRIX Commands

If your \texttt{MATRIX} command is part of a program, it is more likely that you will enter it ‘in line,’ rather than in the matrix calculator. That is as a command in the text editor, in the format,

\[
\texttt{MATRIX ; ... the desired command ... }$
\]

Commands may be entered in this format from the editor, as part of a procedure, or in an input file. All of the applications given elsewhere in this manual are composed of in line commands, as are the examples given in Section R12.1 and in many places in the preceding chapters.

The essential format of a \texttt{MATRIX} command is

\[
\texttt{MATRIX ; name = result ; ... additional commands ... }$
\]

If you wish to see the ‘result’ but do not wish to keep it, you may omit the ‘; name =.’ (The same applies to the scalar calculator described in the next chapter.) For example, you are computing a result and you receive an unexpected diagnostic. We sometimes come across a matrix, say \texttt{rxx}, that we thought was positive definite, but when we try something like \texttt{MATRIX ; Sinv(rxx) $}, a surprise error message that the matrix is not positive definite shows up. A simple listing of the matrix shows the problem. The .001 in the 4,4 element is supposed to be a 1.0. Now we have to go back and find out how the bad value got there – some previous calculation did something unexpected.
Matrix ; Sinv(Rxx) $

Error 185: MATRIX - GINV,SINV,CHOL singular, not P.D. if SINV or CHOL

Matrix ; List ; Rxx $

Matrix Result has 4 rows and 4 columns.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.00000</td>
<td>.66647</td>
<td>.13895</td>
<td>.70138</td>
</tr>
<tr>
<td>2</td>
<td>.66647</td>
<td>1.00000</td>
<td>-.27401</td>
<td>.25449</td>
</tr>
<tr>
<td>3</td>
<td>.13895</td>
<td>-.27401</td>
<td>1.00000</td>
<td>-.05935</td>
</tr>
<tr>
<td>4</td>
<td>.70138</td>
<td>.25449</td>
<td>-.05935</td>
<td>.00100</td>
</tr>
</tbody>
</table>

If you want only to see a matrix, and not operate on it, you can just double click its name in the project window. That will open a window that displays the matrix. The offending $r_{xx}$ matrix shown above is displayed in Figure 9.4.

Matrix results will be mixtures of matrix algebra, i.e., addition, multiplication, subtraction, and matrix functions, such as inverses, characteristic roots, and so on, and, possibly, algebraic manipulation of functions of matrices, such as products of inverses.

### 9.2.3 Matrix Output

The results of MATRIX commands can be matrices with up to 50,000 elements, and can thus produce enormous amounts of output. As such, most of the display of matrix results is left up to your control.

Matrix results are always displayed in the calculator window. When commands are in line, results are generally not shown unless you specifically request the display with ; List. (See Section 9.2.5.) The figures below demonstrate.
When the computed result has more than five columns or more than 20 rows, it will be shown in the output window as a placeholder (object).

![Output Window](image)

**Figure 9.6 Matrix Result in the Output Window**

If you double click the object, you can display the full matrix in a scaleable window.

### 9.2.4 Matrix Results

When **MATRIX** commands are given in line, the default is not to display the results of any matrix computations on the screen or in the output file. It is assumed that in this mode, results are mostly intermediate computations. The output file will contain, instead, a listing of the matrix expression and either a confirmation that the result was obtained or just a statement of the expression with a diagnostic in the trace file. For example, the command

```
MATRIX ; a = Iden(20) $
```

produces only an echo of the expression.

You can request full display of matrices in the output file by placing

```
; List
```

before the matrix to be listed. Note how this has been used extensively in the preceding examples in Section R12.1. This is a switch that will now remain on until you turn it off with **Nolist**. When the end of a command is reached, **Nolist** once again becomes the default. The **Nolist** and **List** switches may be used to suppress and restore output at any point. When the **Nolist** specification appears in a **MATRIX** command, no further output appears until the **List** specification is used. At the beginning of a command, the **List** switch is off, regardless of where it was before. If you are doing many computations, you can suppress some of them, then turn the output switch back on, in the middle of a command. For example:
MATRIX ; Nolist
; xxi = <x'x>
; List
; Root(xxi) $ displays only the characteristic roots of the inverse of a particular $X'X$ matrix. Neither $x'x$ nor $xxi$ are displayed.

Displaying matrices that already exist in the matrix work area requires only that you give the names of the matrices. I.e.,

MATRIX ; List ; abcd ; qed $ Note, separated by semicolons, not commas.

would request that the matrices named $abcd$ and $qed$ be displayed on your screen. You might also want to see the results of a matrix procedure displayed, without retaining the results. The following are some commands that you might type:

MATRIX ; Root(xx) $ lists characteristic roots of $xx$.
MATRIX ; a* b $ displays the matrix product $ab$.
MATRIX ; Mean(x*) $ displays the means of all variables whose names begin with $x$.

These commands just display the results of the computations; they do not retain any new results.

9.2.5 Matrix Statistical Output

Your matrix procedures will often create coefficient vectors and estimated covariance matrices for them. For any vector, $beta$, and square matrix, $v$, of the same order as $beta$, the command

MATRIX ; Stat (beta,v) $ will produce a table which assumes that these are a set of statistical results. The table contains the elements of $beta$, the diagonal elements of $v$ and the ratios of the elements of $beta$ to the square root of the corresponding diagonal element of $v$ (assuming it is positive). For example, the listing below shows how the Stat function would redisplay the model results produced by a POISSON command from one of our earlier examples.

NOTE: MATRIX ; Stat(vector, matrix) $ has no way of knowing that the matrix you provide really is a covariance matrix or that it is the right one for the vector that precedes it. It requires only that ‘vector’ be a vector and ‘matrix’ be a square matrix of the same order as the vector. You must insure that the parts of the command are appropriate.

The routine to produce model output for a matrix computed set of results can be requested to display variable names by adding a namelist with the appropriate variables as a third argument in the MATRIX ; Stat(b,v) $ function. If your estimator is a set of parameters associated with a set of variables, $x$, they are normally labeled $b_1$, $b_2$, etc. Adding the namelist to the MATRIX ; Stat(b,v,x) $ function carries the variable labels into the function. An example follows: (Some results are omitted.)
Results from Poisson regression

| Variable | Coefficient | Standard Error | b/St.Er. | P[|Z|>z] | Mean of X |
|----------|-------------|----------------|---------|--------|-----------|
| Constant | -5.33586673 | .69883851       | -7.635  | .0000  |           |
| C67      | .74916894   | .17622269       | 4.251   | .0000  | 29411765  |
| C72      | .84733072   | .18996097       | 4.461   | .0000  | 14705882  |
| C77      | .36254858   | .24697201       | 1.468   | .1421  | 12415882  |
| P77      | .39897106   | .12475671       | 3.198   | .0014  | 15588235  |
| LOGMTH   | .84714538   | .06903915       | 12.271  | .0000  | 6903915   |

**MATRIX ; Stat(b,varb)**

| Number of observations in current sample = 34 |
| Number of parameters computed here = 10 |
| Number of degrees of freedom = 24 |

| Variable | Coefficient | Standard Error | b/St.Er. | P[|Z|>z] |
|----------|-------------|----------------|---------|--------|
| B_1      | -5.33586673 | .69883851       | -7.635  | .0000  |
| B_6      | .74916894   | .17622269       | 4.251   | .0000  |
| B_7      | .84733072   | .18996097       | 4.461   | .0000  |
| B_8      | .36254858   | .24697201       | 1.468   | .1421  |
| B_9      | .39897106   | .12475671       | 3.198   | .0014  |
| B_10     | .84714538   | .06903915       | 12.271  | .0000  |

**MATRIX ; Stat(b,varb,x)**

| Variable | Coefficient | Standard Error | b/St.Er. | P[|Z|>z] |
|----------|-------------|----------------|---------|--------|
| Constant | -5.33586673 | .69883851       | -7.635  | .0000  |
| C67      | .74916894   | .17622269       | 4.251   | .0000  |
| C72      | .84733072   | .18996097       | 4.461   | .0000  |
| C77      | .36254858   | .24697201       | 1.468   | .1421  |
| P77      | .39897106   | .12475671       | 3.198   | .0014  |
| LOGMTH   | .84714538   | .06903915       | 12.271  | .0000  |

### 9.3 Using MATRIX Commands with Data

*LIMDEP’s* matrix package is designed to allow you to manipulate large amounts of data efficiently and conveniently. Applications involving up to three million observations on 150 variables are possible. With **MATRIX**, manipulation of a data matrix with 1,000,000 rows and 50 columns, which would normally take 400 megabytes of memory just to store, is not only feasible, but no more complicated than it would be if the data set had only 100 rows instead! It is important for you to be aware of how this is done in order to use this program successfully.

The essential ingredient is the form in which matrix results generally appear in econometrics. It is quite rare for an estimator or a procedure to be based upon “data matrices,” per se. Rather, they almost always use functions of those matrices, typically moments, i.e., sums of squares and cross products. For example, an OLS estimator, \( \hat{b} = (X'X)^{-1}X'y \), can be viewed as a function of \( X \) and \( y \). But, it is much more useful to view it as a function of \( X'X \) and \( X'y \). The reason is that, regardless of the number of observations in the data set, these matrices are \( K \times K \) and \( K \times 1 \), and \( K \) is usually small. *LIMDEP* uses this result to allow you to manipulate your data sets...
with matrix algebra results, regardless of the number of observations. To underscore the point, consider that currently, most other econometrics packages provide a means of using matrix algebra. But, to continue our example, in order to do a computation such as that for \( b \) directly, some of them must physically move the data that comprise \( X \) into an entity that will be the matrix, \( X \). Thus \( X \) must be created, even though the data used to make \( X \) are already in place, as part of the data set currently being analyzed. It is this step which imposes the capacity constraints on some econometrics programs. Avoiding it allows \textsc{LIMDEP} to manipulate data matrices of any length. The utility of this approach will be clear shortly.

It is important to keep in mind the distinction between two kinds of matrices that you will be manipulating. We define them as follows:

- **Data matrices**: A data matrix is a set of rows defined by observations and columns defined by variables. The elements of the data matrix reside in your data area.

- **Computed matrices**: A computed matrix is the result of an operation that is based on data matrices or other computed matrices. The elements of a computed matrix will reside in your matrix work area, which is defined below.

The distinction is purely artificial, since, as will soon be evident, every numeric entity in \textsc{LIMDEP} is a matrix. The important element is that the size of a data matrix is \( n \times K \) where \( n \) is the current sample size and \( K \) is a dimension that you will define. The size of a computed matrix is \( K \times L \) where \( K \) and \( L \) are numbers of variables, or some other small values that you will define with your commands.

### 9.3.1 Data Matrices

To use your data to compute matrices, you will usually define ‘data matrices.’ This amounts to nothing more than labeling certain areas of the data array; you do not actually have to move data around (whatever that might mean) to create a data matrix. For \textsc{LIMDEP}’s purposes, a data matrix is any set of variables which you list. You can overlap the columns of data matrices in any way you choose; data matrices may share columns. An example appears below. One useful shortcut which can be used to ‘create’ a data matrix is simply to associate certain variables and observations with the matrix name by using \texttt{NAMELIST}. The variables are defined with the \texttt{NAMELIST} command. The rows or observations are defined by the current sample, with the \texttt{SAMPLE}, \texttt{REJECT/INCLUDE}, \texttt{DRAW}, and \texttt{PERIOD} commands. If you change the current sample, the rows of all existing data matrices change with it. If you change the variables in a namelist, you redefine all matrices that are based on that namelist.

For example, suppose the data array consists of the following:

<table>
<thead>
<tr>
<th>YEAR</th>
<th>CONS</th>
<th>INVEST</th>
<th>GNP</th>
<th>PRICES</th>
</tr>
</thead>
<tbody>
<tr>
<td>1995</td>
<td>1003</td>
<td>425</td>
<td>1821</td>
<td>124.5</td>
</tr>
<tr>
<td>1996</td>
<td>1047</td>
<td>511</td>
<td>2072</td>
<td>139.2</td>
</tr>
<tr>
<td>1997</td>
<td>1111</td>
<td>621</td>
<td>2341</td>
<td>154.7</td>
</tr>
<tr>
<td>1998</td>
<td>1234</td>
<td>711</td>
<td>2782</td>
<td>177.6</td>
</tr>
</tbody>
</table>

Two data matrices, \texttt{demand} and \texttt{alldata} would be defined by the command

```
NAMELIST ; demand = cons,invst,gnp
 ; alldata = year,cons,invst,gnp,prices $
```

Notice that these data matrices share three columns. In addition, any of the 31 possible subsets of variables can be a data matrix, and all could exist simultaneously.
The number of rows each data matrix has depends on the current sample. For example, to have the matrices consist of the last three rows of the data, it is necessary only to define

\[
\text{SAMPLE ; 2 - 4 $}
\]

You can vary the sample at any time to redefine the data matrices. For example, suppose it is desired to base some computations on \textit{demand} using all four years, and then compute other matrices using \textit{alldata} only for the last three years. The sequence might appear as follows:

\[
\begin{align*}
\text{SAMPLE ; All $} \\
\text{MATRIX commands using demand} \\
\text{SAMPLE ; 2 - 4 $} \\
\text{MATRIX commands using alldata}
\end{align*}
\]

The reason for the distinction between data and computed matrices is this: Consider the computation of a matrix of weighted sums of squares and cross products

\[
F = \frac{1}{n}X'WX
\]

where \(X\) is \(n\times K\) with \(n\) being the sample size, and \(W\) is an \(n\times n\) diagonal matrix of weights. Suppose \(n\) were 10,000 and \(K\) were 20. In principle, just setting up \(X\) and \(W\) for this computation would require at least \(8(10000\times 20 + 10000\times 10000)\), or over 800 million bytes of memory, before computation even begins! But, computations of this size are routine for \textit{LIMDEP}, because

- \(F = \frac{1}{n}\sum_{i=1}^{n}w_i x_i x'_i\) where \(x_i\) is a row of \(X\), which is always only \(20\times 20\), and
- The data needed for the sum already exist in your data area.

That is, by treating this sort of computation as a summing operation, not as a formal matrix product, we can achieve tremendous efficiencies. The important feature to exploit is that regardless of \(n\), the result will always be \(K\times K\).

### 9.3.2 Computations Involving Data Matrices

You can manipulate any sized data matrix with \texttt{MATRIX}. There are two simple rules to remember when using large samples:

- Ensure that in any expression, \texttt{MATRIX ; name = result $}, the target matrix (\textit{name}) is not of the order of a data matrix. That is, neither rows nor columns is \(n\). This will be simple to achieve, since the sorts of computations that you normally do will ensure this automatically.
- Ensure that when data matrices appear in an expression, they are either in the form of a moment matrix, i.e., in a summing operation, or they appear in a function that does summing.

Suppose that \(x\) and \(y\) are data matrices defined as above with 500,000 rows and 25 columns each (i.e., they are very large). Any operation that uses \(x\) or \(y\) directly will quickly run into space problems. For example,

\[
\text{MATRIX ; z = x' * y $}
\]

(the matrix product equal to the transpose of \(x\) times \(y\)) is problematic, since copies of both \(x'\) and \(y\) must be created. But, the apostrophe is a special operator, and
can be computed because in this form *LIMDEP* knows that the operation is a sum of cross products, and will be only 25×25. The second rule above, then, amounts to this: When data matrices appear in matrix expressions, they should always be in some variant of \( x'y \), i.e., as an explicit sum, or in one of the special moment functions listed in Section 9.7, such as Xdot. The apostrophe (‘) is a special operator in this setting. Although it can be used in multiplying any matrices, it is the device which allows you to manipulate huge data matrices, as illustrated by the examples given at the beginning of this chapter. All of them work equally well with small samples or huge ones. The commands are all independent of the number of observations.

Finally, consider the weighted sum above, \( F = (1/n)X'WX \) where there are 10,000 observations and 20 variables. Once again, the result is going to be 20×20. *LIMDEP* provides many different ways to do this sort of computation. For this case, the best way to handle it is as follows:

```lml
NAMELIST ; x = the list of 20 variables $
SAMPLE ; ... set up the 10,000 observations $
CREATE ; w = the weighting variable $
MATRIX ; f = 1/n * x'[w]x $
```

This would work with 10 or 10,000,000 observations. The matrix \( f \) is always 20×20.

### 9.4 Manipulating Matrices

The preceding described how to operate the matrix algebra package. The example in Section 9.1 also showed some of the more common uses of **MATRIX**. This and the following sections will now detail the specifics of *LIMDEP*’s matrix language. In addition to the basic algebraic operations of addition, subtraction, and multiplication, *LIMDEP* provides nearly 100 different functions of matrices, most of which can, themselves, be manipulated algebraically. In this section, we will list the various conventions that apply to these operations.

#### 9.4.1 Naming and Notational Conventions

Every numeric entity in *LIMDEP* is a matrix, and you will rarely have to make a distinction among them. For example, in the expression,

```lml
MATRIX ; f = q’r $
```

\( q \) and \( r \) could be any mix of:

- variables,
- data matrices,
- computed matrices,
- named scalars,
- literal numbers, e.g., 2.345,
- the number (symbol) 1, which has special meaning in matrix multiplication.

Of course, the two entities must be conformable for the matrix multiplication, but there is no requirement that two matrices be the same type of entity. (Usually, they will be.) For convenience, we will sometimes make the following definitions:
• Variable names are \textit{vnames}.
• Namelists are \textit{xnames}.
• Computed matrices are \textit{mnames}.
• Scalars are \textit{rnames}.
• Numbers are \textit{scalars}.

At any time, you can examine the contents of the tables of these names in your project workspace, just by clicking the particular name in your project window. Whenever you create an entity in any of these tables, all of the others are checked for conflicts. For example, if you try to create a variable named \textit{q}, and there is already a matrix with that name, an error will occur.

There are two reserved matrix names in \textit{LIMDEP}. The matrix program reserves the names \textit{b} and \textit{varb} for the results of estimation programs. These two names may not appear on the left hand side of a matrix expression. They may appear on the right, however. (This is known as ‘read only.’)

There are a few additional names which are read only some of the time. For example, after you use the \textit{SURE} command, \textit{sigma} becomes a reserved name. Model output will indicate if a reserved name has been created.

In the descriptions of matrix operations to follow,

• \textit{xname} is the name of a data matrix. This will usually be a namelist. However, most data manipulation commands allow you merely to give a set of variable names instead.
• \textit{mname} is the name of a computed matrix.
• \textit{s} is a scalar. It may be a number or the name of a scalar which takes a value.
• A matrix has \( r \) rows and \( c \) columns.
• Matrices in matrix expressions are indicated with boldfaced uppercase letters.
• The transpose of matrix in a matrix algebra expression \textit{C} is denoted \( C^\prime \).
• The apostrophe, \(',\) also indicates transposition of a matrix in \textit{LIMDEP} commands.
• The ordinary inverse of matrix \textit{C} is denoted \( C^{-1} \).

The result of a procedure that computes a matrix \textit{A} is denoted \textit{a}. Input matrices are \textit{c}, \textit{d}, etc. In any procedure, if \textit{a} already exists, it may appear on both sides of the equals sign with no danger of ambiguity; all matrices are copied into internal work areas before the operation actually takes place. Thus, for example, a command may replace \textit{a} with its own transpose, inverse or determinant. You can replace a matrix with some function of that matrix which has different dimensions entirely. For example, you might replace the matrix named \textit{a} with \( a^\prime a \) or with \( a \)’s rank, trace or determinant.

Note in these definitions and in all that follows, we will make a distinction between a matrix expression (in theory), such as \( F = (1/n)X^\prime X \), and the entities that you manipulate with your \textit{LIMDEP} commands, for example, ‘you have created \( f = x^\prime x \).’ There are thus three sets of symbols. We will use bold upper case symbols in matrix algebra descriptions; we will use bold lower case symbols for the parts of \textit{LIMDEP} commands. We will use italic lower case symbols when we refer, outside \textit{LIMDEP} commands, to the names of matrices, variables, namelists and scalars you have created. Consider, for example, the following: ‘The sample second moment matrix of the data matrix \textit{X} is \( F = (1/n)X^\prime X \). You can compute this by defining \textit{X} with a command such as \textbf{NAMELIST ; x = one,age,income $}, then using the command \textbf{MATRIX ; f = 1/n*x'x $}. After you execute this command, you will see the matrix \textit{f} in your project window listing of matrices. The namelist \textit{x} will also appear in the project window list of namelists.’ You might note, we have used this convention at several points above.
9.4.2 Matrix Expressions

Most of the operations you do with matrices, particularly if you are constructing estimators, will involve expressions, products, sums, and functions such as inverses. This section will show how to arrange such mathematical expressions of matrices. We have used these procedures at many points in our earlier discussion.

As noted above, every numerical entity in LIMDEP is a matrix and may appear in a matrix expression. There are very few functions that require data matrices. These will be noted below. The algebraic operators are

* for matrix multiplication,
+ for addition,
- for subtraction,
\(\prime\) (apostrophe) for transposition and also for transposition then multiplication,
/ for a type of division (see below),
\(^\) for raising a matrix to a power (several forms, see below).

Thus, \(c*d\) equals \(C \times D\) and \(c*Ginv(c)\) (or \(c*<c>\)) equals \(C\) times its inverse, or \(I\), and \(c*Ginv(c)*c\) equals \(C\) times its inverse, or \(I\), and

\(c\prime*Ginv(c)*c\) equals \(C\) times its inverse, or \(I\), and

\(c\prime\) is a crucial part of this package.

When scalars appear in matrix computations, they are treated as scalars for purposes of computation, not as matrices. Thus, \(AsB\prime\), where \(s\) is a scalar, is the same as \(sAB\prime\). The 1×1 matrix in the middle does not interfere with conformability; it produces scalar multiplication. 1×1 matrices which are the result of matrix computations, such as quadratic forms, also become scalars for purposes of matrix multiplication. Thus, in \(a' * r'b*r * a\) will not require conformability of \(A'\) and \(r'\) (number columns of \(A'\) equal number of rows of \(r'\)) if the quadratic form \(r'Br\) is collected in one term; also, \(A'*r*B*r*A\) does require conformability, but the same expression could be written \(a' * r'[b]r * a\) to achieve greater efficiency. Also, if \(r\) happens to be a variable, this may be essential. The implications of these different forms will be presented in detail below.

All syntaxes are available for any entity, so long as conformability is maintained where appropriate. \(A\) and \(B\) are any matrix; \(w\) is any vector, row or column, including, if desired, a variable; and, \(C\) is any matrix. (Once again, a matrix is any numeric entity – there is no need to distinguish, e.g., variables from previously computed matrices.)

Each result in the following table produces a result that, for later purposes, may be treated as a single matrix.

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a')</td>
<td>transpose of (a) times (b)</td>
</tr>
<tr>
<td>(a[w]b)</td>
<td>(a'[\text{diag}(w)] b) (Do not create diagonal matrices!)</td>
</tr>
<tr>
<td>(a&lt;\text{row}&gt;b)</td>
<td>(a'[\text{diag}(w)]^{-1} b)</td>
</tr>
<tr>
<td>(a[c]b)</td>
<td>(a' c b)</td>
</tr>
<tr>
<td>(a&lt;c&gt;b)</td>
<td>(a' c^{-1} b) (c) is any matrix.</td>
</tr>
<tr>
<td>(&lt;a&gt;)</td>
<td>(a^{-1})</td>
</tr>
<tr>
<td>([a])</td>
<td>(G^{-2}) inverse of (a).</td>
</tr>
<tr>
<td>(&lt;a'\ b&gt;)</td>
<td>((a' \ b)^{-1})</td>
</tr>
<tr>
<td>(&lt;a[w]\ b&gt;)</td>
<td>((a'[w] \ b)^{-1})</td>
</tr>
<tr>
<td>(&lt;a&lt;\text{row}&gt;\ b&gt;)</td>
<td>((a&lt;\text{row}&gt; \ b)^{-1})</td>
</tr>
</tbody>
</table>

Table 9.1 Matrix Expressions

In a matrix expression, the symbol ‘1’ can be used where needed to stand for a column of ones. Thus,
1’a = a row of ones times matrix a
a’a = transpose of matrix a times a column of ones.

Note that in each of these cases, the apostrophe is an operator that connotes multiplication after transposition.

**NOTE:** You should never need to compute a’ * b. Always use a’b. Thus, in the earlier example, c’< c > c is better than c **Ginv(c)** * c or c’< c >* c.

In any matrix function list, you may use the transpose operator for transposition. For example, two ways to obtain the sum of a matrix and its transpose are

\[
\text{sum} = a + a' \quad \text{and} \quad \text{sum} = \text{Msum}(a, a').
\]

The transpose of a matrix may appear in an expression simply by writing it with a following apostrophe. For example,

\[
a’c’c’ a \quad \text{could be computed with} \quad a’ * c’ * c * a
\]

though a’ * c’c * a would be necessary if c were a data matrix.

You may string together as many matrices in a product as desired. As in the example, the terms may involve other matrices or functions of other matrices. For example, the following commands will compute White’s heteroscedasticity corrected covariance matrix for the OLS coefficient vector.

```plaintext
NAMELIST ; x = list of Rhs variables $
REGRESS  ; Lhs = y ; Rhs = x ; Res = e $
CREATE   ; esq = e^2 $
MATRIX ; white = <x'x> * Bhhh(x,e) * <x'x> $ 
```

The **CREATE** command that computes the squared residuals is actually unnecessary. The last two lines could be combined in

```plaintext
MATRIX ; white = <x'x> * Bhhh(x,e) * <x'x> $ 
```

**LIMDEP** also provides a function to compute the center matrix for the Newey-West estimator;

\[
\text{Nwst}(x,e,l) \quad \text{computes the Newey-West middle matrix for l lags. } l = 0 \Rightarrow \text{White.}
\]

You may also multiply simple matrices that you enter directly. For example

\[
\begin{bmatrix}
1 & 3 \\
3 & 5
\end{bmatrix}
\begin{bmatrix}
2 & 4 \\
5 & 5
\end{bmatrix}
= \begin{bmatrix} 1/3.5 & 2.4/5.5 \end{bmatrix}.
\]

The multiplication operator sorts out scalars or 1 1 matrices in a product. Consider, for example, \( V = A(r'Ar)^{-1}A' \). If \( r \) is a column vector, this is a matrix \((AA')\) divided by a quadratic form. To compute this, you could use

```plaintext
MATRIX ; v = a * <r'[a]r> * a' $ 
```
The scanner will sort out scalars and multiply them appropriately into the product of matrices. But, in all cases, matrices which are not 1×1 must be conformable for the multiplication. Thus, if \( r \) were a matrix instead of a vector, it might not be possible to compute \( V \).

The ‘+’ operator is used to add matrices. Thus, to add the two matrices above instead of multiply them, we could use

\[
\begin{bmatrix} 1 & 3 \\ 3 & 5 \end{bmatrix} + \begin{bmatrix} 2 & 4 \\ 5 & 5 \end{bmatrix} = [1 / 3,5] + [2,4 / 5,5].
\]

The matrix subtraction operator is ‘-’. Thus, \( a - c \) gives \( A - C \) (of course).

You may also combine the +, -, and * operators in a command. For example, the restricted least squares estimator in a classical regression model, when the linear restrictions are \( Rb = q \), is

\[
b_r = b_u - (X'X)^{-1}R'(R(X'X)^{-1}R')^{-1}(Rb_u - q).
\]

This could be computed with

```plaintext
NAMELIST ; x = list of variables $ MATRIX ; bu = <x'x> * x'y ; r = ... ; rt = r' ; q = ... ; d = r * bu - q ; xxi = <x'x> ; br = bu - xxi * r' * <rt'[xxi]r>* * d $
```

(Why did we transpose \( r \) into \( rt \) then use \( rt' \), which is just \( r' \), in the last expression? Because the apostrophe operator is needed to produce the correct matrix multiplication inside the < > operation. There are other ways to do this, but the one above is very convenient.) Notice in the preceding that if there is only one constraint, \( r \) will be a row vector, and the quadratic form will be a scalar, not a matrix.

Any of the product arrangements shown in Table R12.1 may appear in any function or expression as if it were already an existing matrix. For example, \( \text{Root}(<x'x>[w;x]) \) computes the characteristic roots of \([\Sigma_w;x;x]^{-1}\). But, longer matrix expressions may not be grouped in parentheses, nor may they appear as arguments in other matrix functions. Expressions which must be used in later sums and differences or functions must be computed first. There will usually be other ways to obtain the desired result compactly. For examples,

\[
d = \text{Sinv}(a + c) \text{ is invalid but can be computed with } d = \text{Nvsm}(a,c),
\]
\[
d = (a + c) * q \text{ is invalid but can be computed with } d = \text{Msum}(a,c) * q,
\]
and
\[
d = \text{Ginv}(a * q * r) \text{ is invalid, but can be computed with } d = \text{Iprd}(a,q,r).
\]

The functions Msum, Mdif, Nvsm, and Iprd may facilitate grouping matrices if necessary.

9.5 Entering, Moving, and Rearranging Matrices

To define a matrix, use

```plaintext
MATRIX ; name = [... row 1 / ... row 2 ... / ... ] $
```

Elements in a row are separated by commas while rows are separated by slashes. For example,
MATRIX ; a = [1,2,3,4 / 4,3,2,1 / 0,0,0,0] $
creates \ a = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 4 & 3 & 2 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$

To facilitate entry of matrices you can use these two arrangements:

\[ k \mid \text{value} = \text{a } K \times 1 \text{ column vector with all elements equal to value} \]
\[ k_{-} \text{value} = \text{a } 1 \times K \text{ row vector with all elements equal to value} \]

Thus, in the last row above, 0,0,0,0 could be replaced with 4_0.

Symmetric matrices may be entered in lower triangular form. For example,

MATRIX ; a = [1 / 2 , 3 / 4 , 5 , 6] $
creates \ a = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 5 \\ 4 & 5 & 6 \end{bmatrix}$

Matrix elements given in a list such as above may be scalars, or even other matrices and vectors. For example, to compute the column vector, \([y', \theta] = [(1/\sigma)\beta',(1/\sigma)]'\) after fitting a tobit model, you could use

TOBIT ; ...
CALC ; theta = 1/s
MATRIX ; gamma = theta * b ; gt = [gamma / theta] $
Note that the slash used here indicates stacking, not division, and that \(gt\) is a column vector.

**Partitioned Matrices**

A partitioned matrix may be defined with submatrices. For example, suppose \(c1\) is a 5×2 matrix and \(c2\) is 5×4. The matrix \(c=[c1,c2]\) is a 5×6 matrix which can be defined with

MATRIX ; c = [c1 , c2] $
The two matrices must have the same number of rows. Matrices may also be stacked if they have the same number of columns. For example, to obtain

\[ F = \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} \text{ use } f = [c1 / c2]. \]

To obtain

\[ M = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \text{ use } m = [m11,m12 / m21,m22]. \]

Symmetric matrices may be specified in lower triangular form. For example, suppose \(M\) were symmetric, so that \(M_{21} = M_{12}'\). \(M\) could be constructed using

\[ m = [m11 / m21 , m22]. \]
**Matrices with Identical Elements**

If a matrix or vector has all elements identical, use $a = \text{Init}(r, c, s)$. This initializes an $r \times c$ matrix with every element equal to scalar, $s$. This is a way to define a matrix for later use by an estimation program. Example 3 in Section R12.1 shows an application. This method can also be used to initialize a row ($r=1$) or column ($c=1$) vector. Alternatively, you could use $a = [c, s]$ for a row vector or $a = [r|s]$ for a column vector.

**Identity Matrices**

To define an $r \times r$ identity matrix, use $a = \text{Iden}(\text{number of rows})$. It may be useful to use a scalar for the number of rows. For example, suppose that $x$ is the name of a namelist of $K$ variables which will vary from application to application and you will require $a$ to be a $K \times K$ identity matrix, where $K$ is the number of variables in $x$. You can use the following:

```
CALC ; k = Col(x) $
MATRIX ; ik = Iden(k) $
```

The notation $[r]$ produces the same matrix as $\text{Iden}(r)$.

**Equating One Matrix to Another**

Use $a = c$ to equate $a$ to $c$. To equate $a$ to the transpose of $c$, use $a = c'$. You would typically use this operation to keep estimation results. After each model command, the estimated parameter vector is placed in the read only matrix, $b$. Thus, to avoid losing your coefficient vector, you must equate something to $b$.

**Diagonal Elements of a Matrix in a Vector**

The function

$$a = \text{Vecd}(c)$$

creates column vector $a$ from the diagonal elements of the matrix $c$.

**Diagonal Matrix Created from a Vector**

The command

$$a = \text{Diag}(c)$$

creates a square matrix $a$ with diagonal elements equal to those of row or column vector $c$. If $c$ is a square matrix, the diagonal elements of $a$ will be the same as those of $c$ while the off diagonal elements of $a$ will be zero.

**9.6 Matrix Functions**

There are roughly 50 functions that can be combined with the algebraic operators to create matrix expressions. Some of these, such as $\text{Nvsm}()$ are used to combine algebraic results, while others, such as $\text{Root}()$ are specialized functions that produce complex transformations of matrices.

Any of the constructions in Table 9.1 can be used as a stand alone matrix. For example, to obtain the determinant of $(X'WX)^{-1}$, where $W$ is a diagonal weighting matrix, you can use...
Dtrm(<x'[w]x>). Likewise, several such constructions can appear in functions with more than one input matrix. This should allow you to reduce some extremely complex computations to very short expressions.

**Characteristic Roots and Vectors**

\[ a = C\text{vec}(c) \quad \text{- characteristic vectors.} \]

If \( C \) is a \( K \times K \) matrix, \( A \) has \( K \) columns. The \( k \)th column is the characteristic vector which corresponds to the \( k \)th largest characteristic root, ordered large to small. \( C \) must be a symmetric matrix. If not, only the lower triangle will be used.

\[ A = \text{Root}(c) \quad \text{- characteristic roots of a symmetric matrix.} \]

For symmetric matrix \( C \), \( A \) will be a column vector containing the characteristic roots ordered in descending order. For nonsymmetric matrices, use

\[ a = \text{Cxt}(c) \quad \text{- possibly complex characteristic roots of asymmetric matrix.} \]

The characteristic roots of a nonsymmetric matrix may include complex pairs. The result of this function is a \( K \times 2 \) matrix. The first column contains the real part. The corresponding element of the second column will be the imaginary part, or zero if the root is real. The roots are ordered in descending order by their moduli.

You can use Cxt to obtain the dominant root for a dynamic system. Then, the modulus can be obtained with \( \text{CALC} \). Let \( C \) be the relevant submatrix of the structural coefficient matrix in the autoregressive form. Then,

\[
\text{MATRIX} \quad \text{rt} = \text{Cxt}(C) \quad \text{S} \\
\text{CALC} \quad \text{check} = \text{rt}(1,1)^2 + \text{rt}(1,2)^2 \quad \text{S}
\]

\( \text{Cxt}(c) \) gives the same results as \( \text{Root}(c) \) if \( C \) is a symmetric matrix. But, if \( C \) is nonsymmetric, \( \text{Root}(c) \) gives the wrong answer because it assumes that \( C \) is symmetric, and uses only the lower triangle.

It is always possible to obtain the roots of a symmetric matrix. But, certain nonsymmetric matrices may not be decomposable. If this occurs, an error message results.

The Root function can also be used to find the possibly complex roots of a dynamic equation. If \( C \) is a vector of elements \([c_1, c_2, ..., c_Q]\) instead of a symmetric matrix, then \( A = \text{Root}(c) \) reports in a \( K \times 2 \) matrix the reciprocals of the characteristic roots of the matrix

\[
C = \begin{bmatrix}
  c_1 & c_2 & c_3 & \cdots & c_Q \\
  1 & 0 & 0 & \cdots & 0 \\
  0 & 1 & \cdots & \cdots & 0 \\
  \vdots & \vdots & \vdots & \ddots & \vdots \\
  0 & 0 & \cdots & \cdots & 1 \\
\end{bmatrix}
\]

These are the roots of the characteristic equation, \( 1 - c_1 z - c_2 z^2 - \cdots - c_Q z^Q = 0 \), of the dynamic equation

\[ y_t = c_1 y_{t-1} + c_2 y_{t-2} + \cdots + c_Q y_{t-Q} + \text{other terms}. \]

The dominant root of the system is the largest reciprocal reported. If its modulus is larger than one,
the equation is unstable.

**Scalar Functions**

The following always result in a 1×1 matrix:

\[ a = \text{Dtrm}(c) \quad \text{- determinant of square matrix}, \]
\[ a = \text{Logd}(c) \quad \text{- log-determinant of positive definite matrix}, \]
\[ a = \text{Trce}(c) \quad \text{- trace of square matrix}, \]
\[ a = \text{Norm}(c) \quad \text{- Euclidean norm of vector C}, \]
\[ a = \text{Rank}(c) \quad \text{- rank of any matrix}. \]

The rank is computed as the number of nonzero characteristic roots of \( C'C \). To find the rank of a data matrix \( X \) (i.e., several columns of data in a namelist, \( X \)), you could use

\[ c = \text{Rank}(x). \]

However, this may not be reliable if the variables are of different scales and there are many variables. You should use, instead,

\[
\text{MATRIX} \quad ; \quad C = \text{Diag} (X'X) ; \quad C = \text{Isqr}(C) \ast X'X \ast \text{Isqr}(C) ; \quad \text{Rank}(C) \$ \]

### 9.7 Sums of Observations

There is no obstacle to computing a matrix \( X'X \), even if \( X \) has 1,000,000 rows, so long as the number of columns in \( X \) is not more than 225. The essential ingredient is that \( X'X \) is not treated as the product of a \( K \times n \) and an \( n \times K \) matrix, it is accumulated as a sum of \( K \times K \) matrices. By this device, the number of rows, \( n \), is immaterial (except, perhaps, for its relevance to how long the computation will take). Matrix operations that involve \( C'AC \) or \( C'A^{-1}C \) are, as in all cases, limited to 50,000 cells. But, suppose that \( C \) is 5,000×2 and \( A \) is a diagonal matrix. Then, the result is only 2×2, but apparently it cannot be computed because \( A \) requires 25,000,000 cells. But, in fact, only 5,000 cells of \( A \) are needed, those on the principal diagonal. \textit{LIMDEP} allows you to do this computation by providing a vector (in this case, 5,000×1) instead of a matrix, for a quadratic form. Thus, in \( c'[a]c \), if \( a \) is a column or row vector, \textit{LIMDEP} will expand (at least in principle) the diagonal matrix and compute the quadratic form \( C'AC \) as if \( A \) were \( \text{Diag}(a) \). This result will be crucial when \( C \) is a data matrix, \( X \), which may have tens or hundreds of thousands of rows. The second aspect of the computation of matrices that involve your data is that once the data are in place in the data area, in fact, there is no need to create \( A \) or \( \text{Diag}(a) \) at all. The data are just used in place; you need only use variables and namelists by name.

Invariably, when you manipulate data matrices directly in matrix algebra expressions, you will be computing sums of squares and/or cross products, perhaps weighted, but in any event, of order \( K \times K \). The simple approach that will allow you to do so is to ensure that when \textit{xnames} and \textit{vnames} (namelists and variables) appear in matrix expressions, they appear in one of the following constructions, where \( x \) and \( y \) are namelists of variables and \( w \) is a variable: Some data summation functions are listed in Table 9.2.
Using Matrix Algebra

\[ x'x \] = the usual moment matrix.
\[ X'y \] = cross moments.
\[ X'[w]x \] = \( X'\text{diag}(w)X \), weighted sums. \( W \) is a variable. Or, \( X'[w]Y \).
\[ x'<w>x \] = \( X'\text{[diag(w)]}^{-1}X \), weighted by reciprocals of weights.
\[ <x'x> \] = \( (XX')^{-1} \), inverse of moment matrix.
\[ <x'y> \] = \( (XY')^{-1} \), inverse of cross moments, if it exists.
\[ <x'w>x> \] = \( (X'wX)^{-1} \), inverse of weighted moments. Or \( <X'[w]Y> \).
\[ <x'<w>x> \] = \( (X'<w>X)^{-1} \), inverse, weighted by reciprocals of weights.

Table 0.2 Sums of Observations in Matrix Functions

Again, the use of the apostrophe operator here is important in that it sets up the summing operation that allows you to use large matrices. That is, while logically \( x' * y \) is the same as \( x'y \) for LIMDEP’s purposes, they are very different operations. The left hand side requires that copies of \( x \) and \( y \) be made in memory, while the right hand side requires only that the sum of cross products be accumulated in memory.

**TIP:** All sample moments are computed for the currently defined sample. If the current sample includes variables with missing data, you should make sure the SKIP switch is turned on. Missing values in a matrix sum are treated as valid data, and can distort your results. If you precede the MATRIX command(s) with SKIP, then in summing operations, observations with missing values will be ignored.

The operations described here for manipulating data matrices are logically no different from other matrix operations already described. That is, in your expressions, there is no real need to distinguish data manipulations from operations involving computed matrices. The purpose of this section is to highlight some special cases and useful shortcuts.

**Sums, Means, and Weighted Sums of Observations and Subsamples**

To sum the rows of a data matrix, use

\[ ; \text{name} = x'1 \text{ or } x'\text{one} \]

The symbol, \( 1 \) is allowable in this context to stand for a column of ones of length \( n \). This returns a \( K \times 1 \) column vector whose \( k \)th element is the sum of the \( n \) observations for the \( k \)th variable in \( x \). To obtain a row vector instead, use

\[ ; \text{name} = 1'x \text{ or } \text{one}'x \]

Do note, in most applications, this distinction between row and column vectors will be significant. You can obtain a sample mean vector with

\[ ; \text{name} = 1/n * x'1 \]

A matrix function, Mean, is also provided for obtaining sample means, so

\[ ; \text{Mean}(x) = 1/n * x'1 \]

Note that the Mean function always returns a column vector of means, so if you want a row, you must transpose the column after using the function. \((1/n*1'x\) may be more convenient.) The Mean
function provides one advantage over the direct approach. You can use Mean with a list of variables without defining a namelist. Thus, to obtain the means of $z, x, w, \log(k), f21$, you could use

\[
\text{name} = \text{Mean}(z, x, w, \log(k), f21).
\]

To obtain a weighted mean, you can use

\[
\text{name} = \langle 1'w \rangle \times x'w
\]

where $w$ is the weighting variable. Note that this premultiplies by the reciprocal of the sum of the weights. If the weights sum to the sample size, then you can use $1/n$ or $\langle n \rangle$ instead of $\langle 1'w \rangle$. A related usage of this is the mean of a subsample. To obtain a mean for a subsample of observations, you will need a binary variable that equals one for the observations you want to select and zero otherwise. Call this variable $d$. Then,

\[
\text{name} = \langle 1'd \rangle \times x'd
\]

will compute the desired mean. The Mean function also allows weights, so you can use Mean($x, w$) or Mean($x, d$). In order to use this construction, the parameters of the Mean function must be a namelist followed by a variable.

$X_{vcm}(x)$ - covariance matrix for $X$

and $X_{cor}(x)$ - correlation matrix for $X$
This page intentionally left blank.
Chapter 10

Scientific Calculator

10.1 Introduction

You will often need to calculate scalar results. The scientific calculator, **CALC** is provided for this purpose. For example, you can use **CALC** to look up critical points for the normal, t, F, and chi squared distributions instead of searching a table for the appropriate value. (And, **CALC** will give you any value, not just the few in the tables.) Another case will be calculation of a test statistic such as a $t$ ratio or likelihood ratio statistic. **LIMDEP**’s calculator can function like a hand calculator, but, it is an integral part of the larger program as well. Results you produce with the calculator can be used elsewhere, and results you obtain elsewhere, such as by computing a regression, can be used in scalar calculations. The programs listed in the chapters to follow contain numerous examples. The example below is a simple application.

**Example: Testing for a Common Parameter in a Probit Model**

Suppose a sample consists of 1000 observations in 10 groups of 100. The subsamples are observations 1-100, 101-200, etc. We consider a probit model, $y^* = \beta_0 + \beta_1 x + \varepsilon$, observed $y = 1$ if $y^* > 0$ and 0 otherwise. With $\varepsilon \sim N[0,1]$, Prob[$y=1$]=Φ($\beta_0 + \beta_1 x$). We are interested in using a likelihood ratio statistic to test the hypothesis that the same parameters, $\beta_0$ and $\beta_1$, apply to all 10 subsamples against the alternative that the parameters vary across the groups. We also want to examine the set of coefficients.

The first two commands initialize the log likelihood function and define a place to store the estimates.

```
CALC ; lu = 0 ; i1 = 1 $
MATRIX ; slopes = Init(10,2,0) $
```

The following commands define a procedure to compute probit models and sum unrestricted log L.

```
PROC
CALC ; i2 = i1 + 99 $
SAMPLE ; i1 – i2 $
PROBIT ; Lhs = y ; Rhs = one,x $
CALC ; lu = lu + logl ; i1 = i1 + 100 $
MATRIX ; slopes(i,* = b $
ENDPROC
```

Execute the procedure 10 times, resetting the sample each time.

```
EXECUTE ; i = 1,10 $
```

The next two commands compute the restricted log likelihood.

```
SAMPLE ; 1 – 1000 $
PROBIT ; Lhs = y ; Rhs = one,x $ Computes restricted (pooled) logL.
```
Now, carry out the likelihood ratio test.

\[ \text{CALC} \quad ; \quad \text{chisq} = 2 \times (\text{lu} - \logl) \quad ; \quad \text{prob} = 1 - \text{Chi(chisq,18)} \]$  

\text{CALC} plays several roles in this example. It is used to accumulate the unrestricted log likelihood function, \( \text{lu} \). The counters \( i_1 \) and \( i_2 \) are set and incremented to set the sample to 1-100, 101-200, etc. The loop index, \( i \), is also a calculator scalar, and once it is defined, any other command, such as the \text{MATRIX} command above, can use \( i \) like any other number. Finally, the last \text{CALC} command retrieves the log likelihood from the unrestricted model, computes the test statistic, then computes the tail area to the right of the statistic to determine if the hypothesis should be rejected.

### 10.2 Command Input in CALCULATE

\text{CALCULATE} is the same as \text{MATRIX} in the two modes of input. Select \text{Tools:Scalar Calculator} to open the calculator window, shown in Figure 10.1.

![Figure 10.1 Calculator Window](image)

There are two ways to enter commands in the calculator window. You can type \text{CALCULATE} commands in the smaller ‘Expr:’ window. If your command will not fit on one line, just keep typing. At some convenient point, the cursor will automatically drop down to the next line. Only press Enter when you are done entering the entire command. In this mode of entry, you do not have to end your commands with a $.

Alternatively, you can click the \( f_x \) button to open a subsidiary window, the \text{Insert Function} dialog box that provides a menu of matrix and calculator functions (see Figure 10.2). Select \text{Scalar} to display the calculator functions. By selecting a function and clicking \text{Insert}, you can insert a template for the indicated function into your ‘Expr:’ window in the calculator window. You must then change the arguments in the function (e.g., the ‘\( x \)’ in the \( \Phi(x) \) in Figure 10.2) to the entity that you desire. When you have entered your full expression in the window, press Enter to display the command in the lower part of the window, as shown above.

If your command is part of a program, it is more likely that you will enter it in ‘command mode’ or in what we will label the ‘in line’ format. You will use this format in the editing window. That is, in the format,

\[ \text{CALC} \quad ; \quad \ldots \text{the desired result} \ldots \]$  

Commands may be entered in this format from the editor, as part of a procedure, or in an input file. See Figure 10.3. One difference between the calculator window and display in the text editor or the output window is that in the latter, you must include \( ; \text{List} \) in your command to have the result actually displayed. This is the same as \text{MATRIX}. See Section R13.3 for details on the \( ; \text{List} \) Specification.
Scientific Calculator

Figure 10.2 Insert Function Dialog Box for Calculator Functions

Figure 10.3 CALCULATE Command in Text Editor
**CALCULATE** is similar to **CREATE** at this point, except that instead of calculating whole columns of data, you calculate single, or ‘scalar’ values. When you give a name to the result, it is kept in a work area and you can use it later. For example, suppose you wanted to have the value of \( e \) (Euler’s constant) to use later on. You could, for example, calculate \( e = \text{Exp}(1) \). You could then

\[
\text{CREATE} \quad \text{; etotheax} = e^x \quad (a \times x) $
\]

**CALC** is also similar to **MATRIX** in that if you wish to see a value without keeping it you may type the expression without giving it a name, as in

\[
\text{CALC} \quad ; 1 + \pi \times \text{Log}(25) + 2.5 / 1.23 $
\]

or, for the 99% critical value for a two tailed test from the standard normal distribution, \( Ntb(.995) \). (\( Ntb \) stands for Normal table. You also have a \( t \) table, and so on.)

### 10.3 Results from **CALCULATE**

As shown above, when you are in the calculator window, the result of a calculator expression, named or not, is displayed on your screen when it is obtained. When **CALC** commands are given in command mode, the default is not to display the results of any computations in the output window or in the output file if one is open. We assume that in this mode, results are intermediate computations, for example, the increments to the counters in the example in Section R13.1. Commands that you give will be listed in your trace file in all cases and in your output window.

You can request a full display of results both in the output window and in an output file by placing

\[
; \text{List} \\
\]

before the result to be listed. You can turn this switch off with

\[
; \text{Nolist} \\
\]

Thus, the command **CALC** ; \( \text{tailprob} = \Phi(1) \) will create a named scalar, but will not show any visible numerical results. But,

\[
\text{CALC} \quad ; \text{List} ; \text{tailprob} = \Phi(1) $
\]

will show the result on the screen in the output window. Once the end of a command is reached, \( ; \text{Nolist} \) once again becomes the default. The \( ; \text{Nolist} \) and \( ; \text{List} \) switches may be used to suppress and restore output at any point. When the \( ; \text{Nolist} \) specification appears in a **CALC** command, no further output appears until the \( ; \text{List} \) specification is used to restore the listing. At the beginning of a command, the \( ; \text{List} \) switch is off, regardless of where it was before.
To see a result that was computed earlier, there are several ways to proceed. A **CALC** command can simply ‘calculate’ a name. Thus, in the command format, you could just give the command

```
CALC ; List ; tailprob $
```

You may also open the calculator window and just type the name of the scalar you want to see. Finally, when you obtain a named scalar result, it will be added to the project window. (You must ‘open’ the Scalars data group by clicking the .) When the list of scalars is displayed, click any name to display the value at the bottom of the window, in the border. Double clicking a scalar name will open the **New Scalar** dialog box which may also be used to replace the value of that scalar. See Figure 10.4.

---

**10.4 Forms of CALCULATE Commands – Conditional Commands**

The essential format of a **CALCULATE** command is

```
CALC ; name = result ; … additional commands … $ 
```
If you wish to see the ‘result’ but do not wish to keep it, just omit ‘; name =.’ The same applies to the dialog mode in the calculator window. Scalar results will be mixtures of algebraic expressions (addition, multiplication, subtraction, and division), functions, such as logs, probabilities, etc., and, possibly, algebraic manipulation of functions of scalars or expressions.

All calculator commands may be made conditional, in the same manner as CREATE or MATRIX. The conditional command would normally appear

\[
\text{CALC} ; \text{If (logical expression) name = expression }$
\]

The logical expression may be any expression that resolves either to TRUE or FALSE or to a numeric value, with nonzero implying true. The rules for the expression are identical to those for CREATE (see Section R5.2.2) and REJECT (see Section R7.4), as well as MATRIX, and all forms of DO. In this setting, if the condition is true, ‘name’ is computed; if it is false, name is not computed. Thus, if name is a new scalar, and the condition is false, after the command is given, name will not exist. For example,

\[
\text{CALC} ; \text{If (A(1,1) > rsqrd) q = Log(Dtr(sigma)) }$
\]

An entire set of CALCULATE commands can be made conditional by placing a semicolon after the condition, as in

\[
\text{CALC} ; \text{If (condition) ; name = result ; result }$
\]

If the condition is false, none of the commands which follow it are carried out. This form of condition may appear anywhere in a group of CALCULATE commands. This will be most useful in iterative programs to condition your CALC commands.

### 10.4.1 Reserved Names

You can have a total of 50 scalar results stored in your work area. You can obtain a complete list of the names and values assigned to any scalars in the calculator work area by navigating the project window. Fourteen of the scalars are used by the program to save estimation results, and are reserved. The 14 reserved names are

\[
\text{ssqrd, degfrdm, ybar, logl, kreg, sumsqdev, rsqrd, sy, rho, lmda, nreg, theta, s, exitcode}
\]

You can see the reserved scalars in the project window in Figure R13.4. They are the ones marked as ‘locked’ with the gold key icon. These scalars (save for rho – see the hint below) are ‘read only.’ You may not change them with your commands. Most of these results apply to the linear regression model, but values such as ybar, sy, and logl are saved by nearly all models. Scalars lmda and theta will change from model to model, depending on the ancillary parameters in the model. After you estimate a model, you will find these scalars defined automatically with the indicated values. These values can thereafter be used on the right hand side of any command. The final one, exitcode, is an indicator of the success or failure of the most recent estimation command.
HINT: Since it is such a common application, there is an exception to the read only setting of these scalars. The scalar \( \rho \) may be set by a loop control. For example, for scanning in a model of autocorrelation, you might EXECUTE ; \( \rho = 0, 1, .025 \) $. In general \( \rho \) is not a protected name. However, you cannot delete \( \rho \).

### 10.4.2 Work Space for the Calculator

Although there are 50 scalars available, the 14 protected names leave you a total of 36 to work with. If you find yourself running out of room, the command

\[
\text{CALC} \quad ; \text{Delete name, name, …} \$
\]

can be used to clear space. Note that there is no comma or semicolon between the ; Delete specification and the first scalar name. You may also delete scalars that are not reserved in the project window by highlighting their names and pressing the Del key.

### 10.4.3 Compound Names for Scalars

The names of scalars may be indexed by other scalars, in the form \( ssss:iiii \) where ‘\( ssss \)’ is a name and ‘\( iiii \)’ is an integer valued index scalar. For example,

\[
\text{CALC} \quad ; i = 37 ; \text{value} : i = \pi $
\]

creates a scalar named \( \text{value37} \) and assigns it the value \( \pi \). The procedure in the editor window in Figure 10.5 shows how one might use this feature. The data set consists of 10 groups of 20 observations. The procedure computes a linear regression model using each subsample. Then it catches the log likelihood function from each regression, and puts it in a correspondingly named scalar. Thus, the loop index, \( j \), takes values 1,2,…,10, so the scalar names are \( \text{logl}:j = \text{logl1},…,\text{logl10} \).

### 10.5 Scalar Expressions

The rules for calculator expressions are identical to those for CREATE. The rules of algebra apply, with operations \(^\text{^}\) and \(^\text{@}\) (Box-Cox transformation) taking first precedence, * and / next, followed last by + and -. You may also use any of the functions listed below in any expression. This includes the percentage points or critical values from the normal, \( t \), \( F \), and chi squared distributions, sums of sample values, determinants of matrices, or any other algebraic functions. Chapter 9 describes how to obtain matrix results. You may also use an element of a matrix with its subscript enclosed in parentheses in any scalar calculation. Finally, any particular observation on any variable in your data area may also be used in an expression. For example, you might
In evaluating subscripts for variables, the observation refers to rows in the data array, not the current sample. Expressions may also contain any number of functions, other operators, numbers, and matrix elements. A scalar may appear on both sides of the equals sign, with the result being replacement of the original value. For examples:

\[
\text{PROC}
\begin{align*}
\text{CALC} & : i1 - (j-1)*20 + 1 ; i2 - i1 + 19 \\
\text{SAMPLE} & : i1 - i2 \\
\text{REGRESS} & : \text{QuiteLy} ; \text{Lhs} - i \\
\text{CALC} & : \log1:j = \log1 \\
\text{ENDPROC} & \\
\text{EXEC} & : j = 1.10 \\
\end{align*}
\]

\[
\text{Figure 10.5 Procedure with Indexed Scalar Names}
\]

\[
\text{CREATE} : x = \text{some function} \\
\text{CALC} : q = x(21) * \text{sigma}(2,2)
\]

In evaluating subscripts for variables, the observation refers to rows in the data array, not the current sample. Expressions may also contain any number of functions, other operators, numbers, and matrix elements. A scalar may appear on both sides of the equals sign, with the result being replacement of the original value. For examples:

\[
\text{CALC} : \text{varsum} = b(1)^2 * \text{varb}(1,1) + b(2)^2 * \text{varb}(2,2) + 2 * b(1) * b(2) * \text{varb}(1,2) \\
\text{CALC} : \text{messy} = \text{messy}^2/\pi - Gma(.5)/Gma(.1) * \text{Sum(age)}
\]

If it is necessary to change the algebraic order of evaluation, or to group parts of an expression, use parentheses nested to as many levels as needed. For example,

\[
\text{CALC} : \text{func} = (Gma(3) + Gma(5))^3 + ((x + y)/c) * (f + g)
\]

You may also nest functions. For example,

\[
\text{CALC} : q = \text{Log}(\text{Phi}(a1 + a2 * \text{Exp}(a3 + a4 * \text{Gma}(z))))
\]
There are two constants which can be used by name without having been set by you. At all points in the program, the name ‘\(pi\)’ will be understood to be the number \(\pi = 3.14159\ldots\) Note that this will preempt matrices and scalars named \(pi\), so this name should be avoided in other contexts. The name \(pi\) may also appear in \texttt{MATRIX} and \texttt{CREATE} commands, for example,

\[
\text{MATRIX} \quad ; \quad \text{pii} = \text{pi} \times \text{Iden}(5) $
\]
\[
\text{CREATE} \quad ; \quad f = 1/\left(\text{sg} \times \text{Sqr}(2 \times \text{pi})\right) \times \text{Exp}(-0.5 \times ((x - \mu)/\text{sg})^2) $
\]

When you give a \texttt{CALC}, \texttt{MATRIX}, or \texttt{CREATE} command, the name ‘\(n\)’ is always taken to mean the current sample size. You may use \(n\) in any scalar calculation. For example, after you compute a regression, the log likelihood function could be computed using

\[
\text{CALC} \quad ; \quad l = -n/2 \times \left(1 + \text{Log}(2 \times \text{pi}) + \text{Log}((\text{sumsqdev}/n))\right) $ 
\]

NOTE: \(n\) and \(pi\) have the meanings described above everywhere in \texttt{LIMDEP}. Thus, you could use \(pi\) in a list of starting values, as part of a model command, or in \texttt{CALCULATE}.

### 10.6 Calculator Functions

The functions listed below may appear anywhere in any expression. The arguments of the functions can be any number within the range of the function (e.g., you cannot take the square root of -1) as well as matrix elements and names of other scalars. Function arguments may also be expressions, or other functions whose arguments may, in turn, be expressions or other functions, and so on. For example,

\[
z = \text{Log}((\Phi(a_1 + a_2 + \text{Log}(a_2 + (q + r)^2)))
\]

is a valid expression which could appear in a \texttt{CALC} command. The depth of nesting functions allowed is essentially unlimited. When in doubt about the order of evaluation, you should add parentheses to remove the ambiguity. Also, in functions which have more than one argument separated by commas, such as \texttt{Eql(x,y)} (which equals one if \(x\) equals \(y\)), include expression(s) in parentheses. For example,

\[
\text{Eql}( (x+y) , (r+c)^2 ) 
\]

may not evaluate correctly because of the ‘\(x+y\)’ term. But,

\[
\text{Eql}( (x+y) , ((r+c)^2) ) 
\]

will be fine. The supported functions are listed below:

**Basic Algebraic Functions**

- \texttt{Log(x)} = natural log,
- \texttt{Abs(x)} = absolute value,
- \texttt{Sin(x)} = sine,
- \texttt{Tan(x)} = tangent,
- \texttt{Rsn(x)} = arccosine
- \texttt{Ath(x)} = hyperbolic arctangent = \(\frac{1}{2}\ln[(1+x)/(1-x)]\),
- \texttt{Ati(x)} = inverse hyperbolic arctangent = \([\exp(2x)-1] / [\exp(2x)+1]\).
Relational Functions

\[ \text{Eql}(x,y) = 1 \text{ if } x \text{ equals } y, \ 0 \text{ if not,} \]
\[ \text{Neq}(x,y) = 1 - \text{Eql}(x,y), \]
\[ \text{Sgn}(x) = \begin{cases} 1 & \text{if } x > 0, \\ 0 & \text{if } x = 0, \\ -1 & \text{if } x < 0. \end{cases} \]

Critical Points from the Normal Family of Distributions

In each case, when you enter ‘Fcn(\(P,\ldots\))’ where \(P\) is the probability, LIMDEP finds the \(x\) such that for that distribution, the probability that the variable is less than or equal to \(x\) is \(P\). For example, for the normal distribution, \(\text{Ntb}(0.95) = 1.645\). The \(P\) you give must be strictly between 0 and 1.

\[ \begin{align*}
\text{Ntb}(P) & = \text{standard normal distribution,} \\
\text{Ttb}(P,d) & = t \text{ distribution with } d \text{ degrees of freedom,} \\
\text{Ctb}(P,d) & = \text{chi squared with } d \text{ degrees of freedom,} \\
\text{Ftb}(P,n,d) & = F \text{ with } n \text{ numerator and } d \text{ denominator degrees of freedom,} \\
\text{Ntb}(P,\mu,\sigma) & = \text{normal distribution with mean } \mu \text{ and standard deviation } \sigma.
\end{align*} \]

Probabilities and Densities for Continuous Distributions

\[ \begin{align*}
\Phi(x) & = \text{probability that } N[0,1] \leq x, \\
\Phi(x,\mu,\sigma) & = \text{probability that } N[\mu,\sigma] \leq x, \\
N01(x) & = \text{density of the standard normal evaluated at } x \text{ (Note ‘N-zero-one’),} \\
\text{Lgf}(x) & = \log \text{ of standard normal density } = -\frac{1}{2} (\ln 2\pi + x^2). \ Lgf(0) = .918938542, \\
N01(x,\mu,\sigma) & = \text{density of normal}[\mu,\sigma] \text{ evaluated at } x, \\
\text{Tds}(x,d) & = \text{prob}[t \text{ with } d \text{ degrees of freedom } \leq x], \\
\text{Chi}(x,d) & = \text{prob}[\text{chi squared variable with } d \text{ degrees of freedom } \leq x], \\
\text{Fds}(x,n,d) & = \text{prob}[F \text{ with } n \text{ numerator and } d \text{ denominator degrees of freedom } \leq x], \\
\text{Lgp}(x) & = \text{logit probability } = \exp(1)/(1+\exp(x)), \\
\text{Lgd}(x) & = \text{logit density } = \text{Lgp}(x) \times (1 - \text{Lgp}(x)), \\
\text{Lgt}(P) & = \logit \text{ of } x = \log(P/(1-P)) \text{ for } 0 < P < 1, \\
\text{Xpn}(x,\theta) & = \text{prob}[\text{exponential variable with mean } 1/\theta \leq x], \\
\text{Bds}(x,\alpha,\beta) & = \text{prob}[\text{beta variable with parameters } \alpha,\beta \leq x].
\end{align*} \]

Matrix Dimensions

If \(A\) is the name of a matrix,

\[ \text{Row}(A) = \text{number of rows in matrix } A, \]
\[ \text{Col}(A) = \text{number of columns in matrix } A. \]

If ‘\(X\)’ is the name of a namelist, then

\[ \text{Row}(X) = \text{number of observations in current sample } = n, \]
\[ \text{Col}(X) = \text{number of variables in the namelist}. \]
Sample Statistics and Regression Results

The observations used in any of the following are the current sample less any missing observations. For the Sum, Xbr, Var, and Sdv functions of a single variable, missing data are checked for the particular variable. Thus, Xbr(x1) and Xbr(x2) may be based on different observations. You should keep close track of this if your data have gaps or different sample lengths. For the remaining functions, all observations are used without regard to missing data. For example, in the covariance function, LIMDEP uses all data points, so some data may be missing. Be careful using these to prevent the -999s from distorting the statistics.

Sample Moments

For any variable in your data area, or namelist which contains only one variable name, the functions listed below can be used just like any other function, such as Sqr(2). If you wish only to display the statistic, just calculate it. Otherwise, these functions can be included in any expression.

\[
\begin{align*}
\text{Sum}(\text{variable}) & = \text{sum of sample values}, \\
\text{Xbr}(\text{variable}) & = \text{mean of sample values}, \\
\text{Sdv}(\text{variable}) & = \text{standard deviation of sample values}, \\
\text{Var}(\text{variable}) & = \text{variance of sample values}, \\
\text{Xgm}(\text{variable}) & = \text{the geometric mean}; \ Xgm(x) = \exp\left(\frac{1}{n}\sum \log(x_i)\right), \\
\text{Xhm}(\text{variable},h) & = \text{the harmonic mean using parameter } h; \ Xhm(x,h) = \left[\sum x_i^{-h}\right]^{1/h}.
\end{align*}
\]

The summing functions (Sum, Var, Sdv, Xbr) can be restricted to a subsample by including a second variable in the list. If a second variable appears, the function is compute for nonzero values of that second variable. Thus, Sum(variable, dummy) is the sum of observations for which the dummy variable is nonzero. This allows a simple way to obtain a mean or variance in a subset of the current sample.

Covariance and Correlation

For any pair of variables,

\[
\begin{align*}
\text{Cov}(\text{variable},\text{variable}) & = \text{sample covariance}, \\
\text{Cor}(\text{variable},\text{variable}) & = \text{sample correlation}.
\end{align*}
\]

We note, for obtaining the correlation between a continuous variable, \(x\), and a binary variable, \(d\), one would use the ‘biserial’ correlation. It turns out that the biserial correlation is equal to the ordinary, Pearson product moment correlation. So no special function is created for this. Just use

\[
\text{CALC} ; \text{List} ; \text{Cor} (\text{continuous variable } x, \text{binary variable } d) \; \text{S}
\]

to obtain a biserial correlation coefficient.

Order Statistics

\[
\begin{align*}
\text{Med}(\text{variable}) & = \text{median of sample values} \\
\text{Min}(\text{variable}) & = \text{sample minimum} \\
\text{Max}(\text{variable}) & = \text{sample maximum},
\end{align*}
\]
Qnt\( (\text{quantile,variable}) \) = the indicated quantile for the variable.

To locate the minimum or maximum value in the current sample, use

\[
\begin{align*}
\text{Rmn}(\text{variable}) &= \text{observation number where minimum value of variable occurs,} \\
\text{Rmx}(\text{variable}) &= \text{observation number where maximum value of variable occurs.}
\end{align*}
\]

**Dot Products**

For any vector (matrix with one row or column), which we denote \( c \) or \( d \), or variable in your data set, denoted \( x \) or \( y \),

\[
\begin{align*}
\text{Dot}(c,c) &= c'c, \\
\text{Dot}(c,d) &= c'd, \\
\text{Qfr}(c,A) &= c'Ac \quad (A \text{ is a square matrix conformable with } c.).
\end{align*}
\]

Two forms of the Dot function are

\[
\begin{align*}
\text{Dot}(x,x) &= x'x, \\
\text{Dot}(x,y) &= x'y.
\end{align*}
\]

You may also use the simpler form with the apostrophe, and may mix variables and vectors in the function. Thus, if \( x \) and \( y \) are variables, and \( c \) and \( d \) are vectors, all of the following are admissible (assuming they are conformable):

```
CALC ; x'y ; c'y ; Dot(x,y) ; d'd $
```

**Regression Statistics**

The **CALC** command has several functions which allow you to obtain certain regression statistics, such as an \( R^2 \) in isolation from the rest of the least squares computations. In the following, the list of variables in the parentheses is of the form

\[
\begin{align*}
\text{list} &= \text{independent variables, dependent variable.}
\end{align*}
\]

The dependent variable is always given last in a list. As always, if you want a constant term, include \textit{one}. You can use a namelist for the independent variables if you wish, and the wildcard character, *, may be used to abbreviate lists of variable names. The following functions can be computed, where \( X \) is the list of independent variables:

\[
\begin{align*}
\text{Rsq}(X,y) &= R^2 \text{ in regression of variable } y \text{ on } X, \quad R^2 = 1 - \frac{e'e}{\sum (y_i - \overline{y})^2}, \\
\text{Xss}(X,y) &= \text{explained sum of squares,} \\
\text{Ess}(X,y) &= \text{error, or residual sum of squares,} \\
\text{Tss}(X,y) &= \text{total sum of squares,} \\
\text{Ser}(X,y) &= \text{standard error of regression,} \\
\text{Lik}(X,y) &= \text{log likelihood function.}
\end{align*}
\]

**Count for a Panel**

The number of groups in a panel defined by the stratification variable ‘y’ is given by
\[ \text{Ngi}(y) = \text{number of sequences of consecutive identical values of variable } y. \]

This examines the sample of values and counts the number of runs of the same value, assuming that each run defines a stratum. In a sample of 10, if \( i = 1,1,2,2,3,4,4,4 \), the number of runs (groups) is four.

### 10.7 Correlation Coefficients

There are several types of correlation coefficients that one might compute, beyond the familiar product moment measure. The nonparametric measures of rank correlation and of concordance are additional examples. One might also be interested in correlations of discrete variables, which are usually not measured by simple moment based correlations. The following summarizes the computations of several types of correlations with \textit{LIMDEP}. Some of these are computed with \textbf{CALC}, as described earlier, while a few others are obtained by using certain model commands.

**Pearson Product Moment Correlations for Continuous Variables**

For any pair of variables,

\[ \text{Cor}(\text{variable},\text{variable}) = \text{sample correlation}, \]
Chapter 11

Programming with Procedures

11.1 Introduction

Your first uses of LIMDEP will undoubtedly consist of setting up your data and estimating the parameters of some of the models described in the *Econometric Modeling Guide*. The purpose of this chapter is to introduce LIMDEP’s tools for extending these estimators and writing new ones. The programs described in this chapter will also help you make more flexible use of the preprogrammed estimators, such as in testing hypotheses, analyzing specifications, and manipulating the results of the estimation procedures.

11.2 The Text Editor

The tools and methods described in this chapter will make heavy use of the editing features of the program. The various menus described earlier and in the model sections to follow will be of limited usefulness when you are writing your own programs. The text editor will be essential.

11.2.1 Placing Commands in the Editor

LIMDEP’s editing window shown in Figure 11.1 is a standard Windows text editor. Enter text as you would in any other Windows based text editor. You may enter as much text as you like on the editing screen. The Edit menu provides standard editing features such as cut, copy, paste, go to, and find.
The Insert menu shown in Figure 11.2 can also be used in the editing window. The Insert menu allows you to place specific items on the screen in the editing window:

- **Insert: Command** will place a specific *LIMDEP* command (verb) at the insertion point (where the cursor is). A dialog box allows you to select the verb from a full listing (with explanation) of the verbs.

- **Insert: File Path** will place the full path to a specific file at the insertion point. Several *LIMDEP* commands use files. The dialog box will allow you to find the full path to a file on your disk drive, and insert that path in your command.

- **Insert: Text File** will place the full contents of any text file you select in the editor at the insertion point. You can merge input files, or create input files, using this tool.

### 11.2.2 Executing the Commands in the Editor

When you are ready to execute commands, highlight the ones you wish to submit. Then, you can execute the commands in one of two ways:

- Click the GO button on the *LIMDEP* toolbar. (If the toolbar is not displayed click the Tools:Options/View tab, then turn on the Display Tool Bar option. See Figure 11.3.)

- Select the Run menu at the top of your screen. See Figure 11.4. When commands are highlighted, the first two items in this menu will be:
  - Run selection to execute the selected commands once.
  - Run selection Multiple Times to open a dialog box to specify the number of times to run the highlighted commands.

**TIP:** If you have not selected any lines in the editor, the two selections in the Run menu will be Run Line and Run Line Multiple Times. In this case, the line in question is the line where the cursor is located.
Figure 11.3  Tools: Options/View Menu to Set Up Desktop

Figure 11.4  Run Menu
11.3 Procedures

*LIMDEP* operates primarily as an ‘interpreter.’ This means that commands are submitted one at a time, and carried out as they are received. This is as opposed to a ‘compiler’ which would assemble a number of commands in some fashion, translate them into its own language, then execute them all at once. ‘Batch’ mode, or batching commands provides a middle ground between these, whereby you can submit groups of commands from input files or as streams of commands from the editor or in a procedure. If the use of this is merely to submit a sequence of commands with a small number of keystrokes (for which *LIMDEP* provides several methods), then batching provides nothing more than a convenience. But, *LIMDEP* also provides batch like capabilities which make it operate more like a compiler than an interpreter. Consider the logic of an iterative program:

**Step 1.** Initial setup.
**Step 2.** Compute a result based on current information and previous results.
**Step 3.** Decide whether to exit the iteration or return to Step 2, and act on the decision.

In order to carry out such a sequence of commands, you must have several capabilities available. First, results of Steps 1 and 2 must be retrievable. Second, it must be possible not only to submit the set of commands in Step 2 in a batch mode, it must be possible to do so repeatedly. Step 3 may call for many repetitions of the same set of commands. Here is a trivial example:

**Step 1.** `CALCULATE ; i = 0 $`
**Step 2.** `CALCULATE ; List ; i = i + 1 $`
**Step 3.** If $i \leq 10$, go to Step 2.

If we execute this program, it will display the numbers 1 to 10. The problem of retrievability is obviously solved, assuming, of course, that `CALC` can compute and define something called ‘$i$’ in such a way that later on, $i$ will exist. (Certainly it can; see the previous chapter.) The second step will be carried out 10 times. Obviously, you could simply be the program. That is, type the command and look at $i$. If $i$ is less than or equal to 10, type it again. The point of this discussion is to devise a way to make *LIMDEP* do the repetitions for you.

As noted, *LIMDEP* provides several methods of batching commands. The example above could be handled as follows:

```
CALC    ; i = 0 $
PROCEDURE
    CALC    ; i = i + 1 $
ENDPROCEDURE
EXECUTE ; n = 10 $
```

This example initializes $i$, stores the updating command, then executes the stored command 10 times. There are other ways to do this, as well. For example, a shorter way to display the numbers from 1 to 10 is

```
PROCEDURE
    CALC    ; List ; i $
ENDPROCEDURE
EXEC    ; i = 1,10 $
```

Yet another way to proceed would program the steps literally. This would be
This procedure is only executed once, but it contains a loop within it. It displays, then updates i 10 times.

The device used in each case (and generally) will be the ‘procedure.’ Procedures such as these provide a convenient way to store commands. The EXECUTE command offers numerous options for how to carry out the procedure and how to decide to exit from the procedure.

LIMDEP is highly programmable. As shown in numerous examples already, and throughout the Econometric Modeling Guide, you can arrange long sequences of commands to perform intricate analyses. Procedures, which are similar to small programs greatly extend this capability. Procedures will allow you to automate new estimators that are not already present in LIMDEP, and to compute certain test statistics that are not routine parts of the standard output. The remainder of this chapter will show you how to write and execute procedures.

11.4 Defining and Executing Procedures

To store a set of commands you begin with the command

PROCEDURE or just PROC

This tells LIMDEP that the commands that will follow are not to be executed at the time, but just stored for later use. The end of a procedure is indicated with

ENDPROCEDURE or just ENDPREC

Once a set of commands has been entered as a procedure, you can execute it with

EXECUTE or just EXEC

The EXECUTE command has a number of options which are discussed below.

A procedure can be entered at any point, just by submitting it from the editing window. For example

CREATE ; x = Rnn(0,1) ; y = x + 1 + Rnn(0,2) $  
PROC  
SAMPLE ; first – last $  
REGRESS ; Lhs = y ; Rhs = one,x $  
ENDPROC  
CALC ; first = 1 ; last = 10 $  
EXEC
At the time the procedure is created, the sample limits might not exist. The procedure is defined, the sample limits are set, and, finally, the procedure is executed. The procedure, in turn, sets the sample and computes a regression.

You can also load a procedure from an input file. The file must contain the command `PROCEDURE` at the point at which the procedure is to begin, and `ENDPROCEDURE` at the end of the procedure. These might be the first and last commands in the file if you want only to input a procedure. If you `OPEN` such an input file, it will simply be loaded into the procedure buffer, exactly as if you had typed it. But, remember, the `PROCEDURE` cannot `OPEN` any files itself.

Some notes on procedures:

- The procedure loader is not a compiler. The commands you type are not checked in any way for validity. If you type nonsense, `LIMDEP` will dutifully store it for you. The problems will show up when you try to execute the procedure. (But, see below, procedures can be edited.)

- A procedure may consist of no more than 2,500 nonblank characters. When the commands are stored, the embedded blanks are removed and comments are stripped off. Still, it may pay to use short names and always use the four letter convention for model commands.

- The procedure may contain up to 50 commands, but remember that you can combine many `CREATE`, `CALC`, or `MATRIX` operations in a single command by separating them with semicolons.

- Only one active procedure can be defined at a time. If you issue a `PROCEDURE` command, any procedure which existed before is immediately erased. But, you can store up to 10 more procedures in a library, which is described in the next section.

- Project files (.LPJ files) always contain not only the active procedure, but also any procedures that you have stored in your procedure library. They become part of the project.

### 11.4.1 Executing a Procedure Silently

Procedures are often used to produce a final result with many intermediate computations. You can suppress intermediate output with

```
EXECUTE ; Silent $
```

This suppresses all output. When the procedure is completed, the `SILENT` switches are turned off. You can then use `MATRIX`, `CALC`, or whatever other means are necessary to inspect the desired final result from the procedure. You might use this in an experiment in which you fit the same model many (possibly thousands of) times and accumulate a statistic from the execution. For example, you might investigate whether the mean of a certain statistic is zero with the following procedure. The procedure is general – you could replace the application specific part with some particular estimation problem. It accumulates a result, then uses the central limit to test the hypothesis that the statistic being computed is drawn from a distribution with mean zero.

```
CALC ; meanb = 0 ; sb = 0 ; nrep = 1000 $
PROC ...
... generate the data set for the model command that computes the statistic
CALC ; meanb = meanb + the statistic
; sb = sb + the statistic ^2 $
ENDPROC
```
The procedure estimates the same model 1,000 times. The statistic of interest is $z$, computed at the last line. Since the model results are not useful, we use ; Silent to suppress them. The number of repetitions is specified generically in a scalar named nrep, so if a larger or smaller sample is desired, it is necessary only to change the fixed value in the first line.

### 11.4.2 Parameters and Character Strings in Procedures

Procedures may have parameters. Define the procedure as follows:

```plaintext
PROC = name (parameter1, …, up to 15 parameters) $
```

Then,

```plaintext
EXECUTE ; Proc = name (actual1, …) $
```

The actual arguments are substituted for the dummy parameters at execution time. This is like a subroutine call, but more flexible. For execution, the passed parameters are simply expanded as character strings, then the procedure, after creation in this fashion, becomes the current procedure. For example,

```plaintext
PROC = Dstats (x) $
DSTAT ; Rhs = x $
ENDPROC
NAMELIST ; zbc = … a list of variables
 ; q123 = a different list of variables $
EXEC ; Proc = Dstats (zbc) $
EXEC ; Proc = Dstats (q123) $
```

Any string may be substituted anywhere in the procedure. This will allow great flexibility. For example, even models can be changed with the procedure.

```plaintext
PROC = Modeler (model) 
Model ; Lhs = y ; Rhs = one,x $
ENDPROC 
EXEC ; Proc = Modeler (probit) $
EXEC ; Proc = Modeler (logit) $
```

- Your procedures may have up to 15 parameters in the list.
- The number of parameters in the EXEC command is checked against the number in the procedure definition at execution time. But, it is not possible to check the consistency of the parameters in the two lists. Thus, you can’t be prevented from sending a bad command to the Modeler routine above.

With this device, you are free to pass variables, namelists, matrices, model names, or any other entities you require. Also, strings can vary in type from one execution to another, though you must be careful to avoid causing conflicts. For example, assuming that $t$ is a scalar in the named procedure, one might use
PROC = name (t) $
$
then,

EXEC ; Proc = name (x) $
EXEC ; Proc = name (1.2345) $

which would not cause a conflict.

Such procedures would generally be useful for creating prepackaged subroutines. For example, the following procedure computes LM statistics for a given model using two sets of variables:

PROC = Lmtest (model, y, x1, x2) $
Model ; Lhs = y ; Rhs = x1 $
MATRIX ; k2 = Col(x2) 
; b2 = k2 _ 0 $
Model ; Lhs = y ; Rhs = x1,x2 
; Start = b, b2 ; Maxit = 0 $
ENDPROC

You could execute this with something like the following:

NAMELIST ; v1 = one,v1a,ddd 
; v2 = ll,g123 $
EXEC ; Proc = Lmtest (probit, y, v1, v2) $
PART II  Econometric Models and Statistical Analyses

Chapter 12  Econometric Model Estimation
Chapter 13  Describing Data
Chapter 14  The Linear Regression Model
Chapter 15  Models for Discrete Choice
Chapter 16  Censoring and Sample Selection
This page intentionally left blank.
Chapter 12
Econometric Model Estimation

12.1 Introduction

The primary function carried out by LIMDEP is the estimation of econometric models. The first part of the documentation, the Reference Guide describes how to use LIMDEP to read a data set, establish the current sample, compute transformations of variables, and carry out other functions that get your data ready to use for estimation purposes. Several important tools, such as the matrix algebra program, scientific calculator and program editor are described there as well. This second part, the econometric modeling guide, will describe some specific modeling frameworks and instructions to be used for fitting these models.

The organization of this manual is by estimation framework, not by model command. We have found that users prefer that the program documentation be oriented toward the types of functions they want to perform, not to an alphabetical listing of commands. As such, you will find the arrangement of topics in this manual rather similar to the arrangement of topics in treatises in econometrics, such as Greene (2008). We begin with descriptive statistics in Chapter 13, various linear regression models in Chapters 14 and 15, and so on.

12.2 Econometric Models

This manual is devoted primarily to the methods by which you can use LIMDEP to fit equations to data and test hypotheses about the relationships implied by that estimation process. For purposes of documenting the program, we use the term ‘model estimation’ broadly, to encompass all those functions that involve manipulation of data to produce statistics to summarize the information the data contain. Thus, this manual begins with a chapter about computing descriptive statistics, which one might not normally consider model building. However, as data summaries, for program purposes, we consider these part of the model building functions in LIMDEP.

The definition of a ‘model’ in LIMDEP consists of the modeling framework, the statement of the variables in the model, and what role the variables will play in that model. The remainder of this chapter will describe in general terms how to use this format to construct model estimation commands in LIMDEP.

12.3 Model Commands

LIMDEP’s model commands all use the same format. The essential parts are as follows:

Model name ; model variables specification ; essential specifications for some models ; optional specifications $
so different models for count data, most of which are extensions of the basic Poisson regression model.

The ‘model variables specification’ generally defines the dependent and independent variables in a model. In almost all cases, the model will include one or more dependent variables, denoted a Lhs, or ‘left hand side’ variable in LIMDEP’s command structure. Independent variables usually appear on the Rhs, or ‘right hand side,’ of a model specification. To continue our example, a Poisson model might be specified using

\[ \text{POISSON } \quad \text{; Lhs = patents ; Rhs = one, r_and_d $} \]

which specifies one of the most well known applications of this model in economics. (The variable ‘one’ is the constant term. We’ll return to this below.) Some ‘model’ commands will have only one of these two specifications, such as

\[ \text{DSTAT } \quad \text{; Rhs = patents $} \]

which requests descriptive statistics for the variable patents. As can be seen here, we use the term ‘model command’ broadly to indicate analysis of a set of data, whether for description or parameter estimation. Other model commands might have only a Lhs variable, such as

\[ \text{SURVIVAL } \quad \text{; Lhs = failtime $} \]

which requests a nonparametric (life table) analysis of a variable named failtime. There are also many other types of variable specifications, such as

\[ \text{; Inst = a set of variable names} \]

which will be used to specify the set of instrumental variables in the 2SLS command.

Most models can be specified with nothing more than the model name and the identification of the essential variables. But, some models require additional specifications in order to be identified. For example, the specific model you want may be a particular case of a broad class of models and in order to specify it, you must provide the ‘essential’ specifications. For example, the basic command for survival modeling (with covariates to provide the ‘model’) would be

\[ \text{SURVIVAL } \quad \text{; Lhs = failtime ; Rhs = one, usehours $} \]

This form of the command is for Cox’s proportional hazard model. In order to fit a parametric model, such as the Weibull model, you would use

\[ \text{SURVIVAL } \quad \text{; Lhs = failtime ; Rhs = one, usehours ; Model = Weibull $} \]

Note the last specification. This is the only way to specify a Weibull model, so for this model, this specification is essential. The Weibull model is requested as a type of survival model by this command. Obviously, not all models have mandatory specifications – the examples above do not. But, many do. The documentation in the chapters to follow will identify these.

Finally, all model frameworks have options which either extend the model itself or control how the model is estimated or how the estimation results are displayed. For example, the following fits a linear regression model and requests a robust estimator of the covariance matrix of the estimates:
REGRESS ; Lhs = profit ; Rhs = one, sales ; Heteroscedasticity consistent $ 

The latter specification does not change the model specification, it requests an additional computation, the White heteroscedasticity consistent estimator. For another example,

REGRESS ; Lhs = profit ; Rhs = one, sales ; Plot residuals $ 

fits a linear model and then plots the residuals. If the latter specification is omitted, the residuals will not be plotted.

12.4 Command Builders

In what follows, we will describe the model commands that you can use to fit the indicated model. In all cases, there are command builders that can be used instead of the text editor and command language. The command builders are described in Sections 4.4, 6.5.4 and 8.2.1.

12.5 Model Groups Supported by LIMDEP

A few of the various model commands and modeling frameworks supported by LIMDEP and NLOGIT are discussed in the chapters to follow. To suggest the breadth of the full menu, the following are the model names for the different classes of estimators supported by the program. Some, such as HISTOGRAM and BURR are quite narrow, single purpose instructions, while others, such as POISSON, call for large classes of models that may (as in this case) contain a large number of different variants. The specific models that are presented in Chapters 13 – 16 in this manual are highlighted in the list. The documentation below will present the basic forms of each of these. Additional specifications, options and model forms are developed in the full manual for LIMDEP.

Descriptive Statistics

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLASSIFY</td>
<td>Discriminant analysis – classification into latent groups.</td>
</tr>
<tr>
<td>DSTAT</td>
<td>Descriptive statistics.</td>
</tr>
<tr>
<td>TABLES</td>
<td>Descriptive statistics for stratified data.</td>
</tr>
<tr>
<td>CROSSTAB</td>
<td>Cross tabulations for discrete data.</td>
</tr>
<tr>
<td>HISTOGRAM</td>
<td>Histograms for discrete and continuous data.</td>
</tr>
<tr>
<td>KERNEL</td>
<td>Kernel density estimation of the density for a variable.</td>
</tr>
<tr>
<td>IDENTIFY</td>
<td>Descriptive statistics (ACF, PACF) for time series data.</td>
</tr>
<tr>
<td>SPECTRAL</td>
<td>Spectral analysis of a time series.</td>
</tr>
</tbody>
</table>

Plotting

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FPLOT</td>
<td>Function plot for user specified function.</td>
</tr>
<tr>
<td>MPLOT</td>
<td>Scatter plot of matrices.</td>
</tr>
<tr>
<td>PLOT</td>
<td>Scatter or time plots of variables against each other.</td>
</tr>
<tr>
<td>SPLOT</td>
<td>Simultaneous scatter plots for several variables.</td>
</tr>
</tbody>
</table>
### Linear Regressions and Variants

#### Single Equation

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FRONTIER</td>
<td>Stochastic frontier models.</td>
</tr>
<tr>
<td>HREG</td>
<td>Heteroscedastic linear regression</td>
</tr>
<tr>
<td>QREG</td>
<td>Quantile regression</td>
</tr>
<tr>
<td>REGRESS</td>
<td>Linear regression models (also OLSQ and CRMODEL).</td>
</tr>
<tr>
<td>TSCS</td>
<td>Time series/cross section, covariance structure models.</td>
</tr>
<tr>
<td>2SLS</td>
<td>Two stage (instrumental variable) estimation of linear models.</td>
</tr>
</tbody>
</table>

#### Multiple Equation

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SURE</td>
<td>Linear seemingly unrelated regression models.</td>
</tr>
<tr>
<td>3SLS</td>
<td>Three stage (IV, GLS) estimator for systems of linear equations.</td>
</tr>
</tbody>
</table>

#### Sample Selection Models

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MATCH</td>
<td>Propensity score matching to analyze treatment effects.</td>
</tr>
<tr>
<td>SELECT</td>
<td>Sample selection models with linear and tobit models.</td>
</tr>
<tr>
<td>INCIDENTAL</td>
<td>Incidental truncation (selection) model.</td>
</tr>
<tr>
<td>SWITCH</td>
<td>Switching regression models.</td>
</tr>
</tbody>
</table>

### Nonlinear Regression, Optimization, Manipulation of Nonlinear Functions

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARMAX</td>
<td>Box Jenkins ARMA and dynamic linear equations.</td>
</tr>
<tr>
<td>BOXCOX</td>
<td>Regression based on the Box-Cox transformation of variables.</td>
</tr>
<tr>
<td>NLSQ</td>
<td>Nonlinear least squares for nonlinear regression models.</td>
</tr>
<tr>
<td>NLSURE</td>
<td>Nonlinear systems of equations, SURE or GMM estimation.</td>
</tr>
</tbody>
</table>

### Analysis of Nonlinear Functions

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FINTEGRATE</td>
<td>Function integration for user specified nonlinear function.</td>
</tr>
<tr>
<td>GMME</td>
<td>GMM estimation of model parameters.</td>
</tr>
<tr>
<td>MAXIMIZE</td>
<td>Maximization of user specified functions.</td>
</tr>
<tr>
<td>MINIMIZE</td>
<td>User defined minimization command.</td>
</tr>
<tr>
<td>WALD</td>
<td>Standard errors and Wald tests for user specified nonlinear functions.</td>
</tr>
</tbody>
</table>

### Single Equation Models for Binary, Ordered and Multiple Discrete Choices

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BINARY CHOICE</td>
<td>Simulation program for all binary choice estimators.</td>
</tr>
<tr>
<td>BIVARIATE</td>
<td>Bivariate probit models, partial observability models.</td>
</tr>
<tr>
<td>BURR</td>
<td>Burr model for binary choice.</td>
</tr>
<tr>
<td>CLOGIT</td>
<td>Multinomial logit model for discrete choice among multiple alternatives</td>
</tr>
<tr>
<td>COMPLLOG</td>
<td>Complementary log log model for binary choice.</td>
</tr>
<tr>
<td>COMPTZ</td>
<td>Gompertz model for binary choice.</td>
</tr>
<tr>
<td>LOGIT</td>
<td>Binary and multinomial choice models based on the logistic distribution.</td>
</tr>
<tr>
<td>MLOGIT</td>
<td>Multinomial logit model.</td>
</tr>
<tr>
<td>MPROBIT</td>
<td>Multivariate probit model.</td>
</tr>
<tr>
<td>MSCORE</td>
<td>Maximum score semiparametric estimation for binary dependent variable.</td>
</tr>
<tr>
<td>NREG</td>
<td>Nonparametric regression models.</td>
</tr>
<tr>
<td>ORDERED</td>
<td>Ordered probability models for ordered discrete choice.</td>
</tr>
<tr>
<td>PROBIT</td>
<td>Several forms of binary choice models.</td>
</tr>
</tbody>
</table>
SEMIPAR  Klein and Spady semiparametric estimator for binary choice.

**Models for Count Data**

**GAMMA**  Gamma model for count data.
**NEGBIN**  Negative binomial regression model.
**POISSON**  Models for count data.

**Models for Censored Variables**

**BTOBIT**  Bivariate tobit models.
**GROUPED**  Regression models for categorical censored data.
**MIMIC**  Multiple indicators and multiple causes for a latent variable.
**NTOBIT**  Nested tobit models.
**TOBIT**  Censored regression models.

**Models for Variables with Limited Range of Variation**

**LOGLINEAR**  Loglinear models, beta, gamma, Weibull, exponential, geometric, inverse Gaussian.
**LOGNORMAL**  Lognormal regression model.
**TRUNCATE**  Truncated regression models.

**Models for Survival Times and Hazard Functions**

**SURVIVAL**  Survival (hazard function) models.

**12.6 Common Features of Most Models**

There is wide variety in the components and features across the different model classes. But, there are several elements that are common to nearly all of them. Some of those are listed below.

**12.6.1 Controlling Output from Model Commands**

These two settings control model results that are generally not reported unless requested.

; Margin  requests display of marginal effects.
; Printvc  requests display of the estimated asymptotic covariance matrix.
 Normally, the matrix is not shown in the model results.

**12.6.2 Robust Asymptotic Covariance Matrices**

These settings control how the covariance matrix for the coefficient estimator is estimated.

; Cluster=spec  requests computation of the cluster form of covariance the estimator.
; Str = spec  is used with ; Cluster to specify a stratified, two level form of data clustering.
; Robust  requests a ‘sandwich’ estimator or robust covariance matrix for several discrete choice models.
12.6.3 Predictions and Residuals

Fitted values (predictions) and residuals from most single equation models are requested as follows:

; List displays a list of fitted values with the model estimates.
; Keep = name keeps the fitted values as a new (or replacement) variable in the data set.
; Res = name keeps the residuals as a new (or replacement) variable.
; Prob = name saves the probabilities as a new (or replacement) variable for discrete choice models such as probit or logit.
; Fill requests that missing values or values outside the estimating sample be replaced by fitted values based on the estimated model.
Chapter 13
Describing Sample Data

13.1 Introduction

This chapter describes methods of obtaining descriptive statistics for one or more variables in your data set. Procedures are given for cross sections and for panel data.

13.2 Summary Statistics

The primary command for descriptive statistics is

\[ \text{DSTAT} \quad ; \quad \text{Rhs} = \text{list of variables} \]

This produces a table which lists for each variable, \( x_k, k = 1, \ldots, K \), the basic statistics:

- **Sample mean**: \( \bar{x}_k = \frac{1}{N_k} \sum_{i=1}^{N_k} x_{ik} \)
- **Standard deviation**: \( s_k = \sqrt{\frac{1}{N_k - 1} \sum_{i=1}^{N_k} (x_{ik} - \bar{x}_k)^2} \)
- **Maximum value**,
- **Minimum value**,
- **Number of valid (nonmissing) cases**.

Use

\[ \text{DSTAT} \quad ; \quad \text{Rhs} = \text{list of variables} \quad ; \quad \text{All} \]

To request, in addition, the skewness and kurtosis measures,

- **Sample skewness**: \( m_3 = \frac{\sum_{i=1}^{N_k} (x_{ik} - \bar{x}_k)^3 / (N_k - 1)}{s_k^3} \)
- **Sample kurtosis**: \( m_4 = \frac{\sum_{i=1}^{N_k} (x_{ik} - \bar{x}_k)^4 / (N_k - 1)}{s_k^4} \)

After the table of results is given, you may elect to display a covariance or correlation matrix (or both) for the variables. The request is added to the command:

- \( \text{Output} = 1 \) to obtain the covariance matrix,
- \( \text{Output} = 2 \) to obtain the correlation matrix,
- \( \text{Output} = 3 \) for both covariance and correlation matrices.
By adding

; All

to the command, we obtain the expanded results that include the skewness and kurtosis measures.

13.2.1 Weights

Weights may be used in computing all of the sums above by specifying

; Wts = name of weighting variable

Weights are always scaled so they sum to the current sample size. Thus, for example, the weighted mean would be

\[ \bar{x}_k = \frac{1}{N_k} \sum_{i=1}^{N_k} w_{ik} x_{ik} \]

where

\[ w_{ik} = \frac{n_k z_{ik}}{\sum_{i=1}^{N_k} z_{ik}} \]

and \( z_{ik} \) = the weighting variable.

13.2.2 Missing Observations in Descriptive Statistics

In all cases, weighted or otherwise, sums are based on the valid observations. DSTAT automatically selects out the missing data. Most other models (save for those in NLOGIT 4.0 and most of the panel data estimators) do not routinely do so unless you have the SKIP switch set. Each variable may have a different number of valid cases, so the table of results gives the number for each one.

| NOTE: The covariance and correlation matrices are based on the subset of observations for which there were no missing data for any variables. Each row in the table of results will list the number of valid cases used for that particular variable. Unfortunately, if different observations are missing for the various variables used in a covariance or correlation matrix, the union of the observations for which all variables are present can contain very few observations. For better or worse, this union is the set of observations used in computing the matrices. |

If your data contain missing values, the scaling in the previous section is automatically adjusted for each variable. Moments are scaled by the number of valid observations or sum of weights for that variable.

13.2.3 Sample Quantiles

You may obtain more detailed statistics about variables by requesting the sample quantiles. This feature produces sample order statistics and the deciles and quartiles of the sample of values for each variable. The keyword in the command is

; Quantiles
Describing Sample Data

For example:

```
CALC ; Ran(12345) $
SAMPLE ; 1 - 1000 $
CREATE ; x = Rnn(0,1) $
CREATE ; y = Rnn(0,1) ^ 2 $
DSTAT ; Rhs = x, y ; Quantiles ; Matrix $
```

The CREATE commands produce random samples from the standard normal and the chi squared[1] distributions. The following output results:

**Descriptive Statistics**
All results based on nonmissing observations.

```
<table>
<thead>
<tr>
<th>Variable</th>
<th>Mean</th>
<th>Std.Dev.</th>
<th>Minimum</th>
<th>Maximum</th>
<th>Cases</th>
</tr>
</thead>
<tbody>
<tr>
<td>All observations in current sample</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>-.177875E-01</td>
<td>.972540</td>
<td>-3.14333</td>
<td>3.02263</td>
<td>1000 0</td>
</tr>
<tr>
<td>Y</td>
<td>.983443</td>
<td>1.42493</td>
<td>.16230E-07</td>
<td>11.6616</td>
<td>1000 0</td>
</tr>
</tbody>
</table>
```

**Order Statistics for Variables**

```
<table>
<thead>
<tr>
<th>Percentile</th>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min.</td>
<td>-3.14333</td>
<td>.16230E-07</td>
</tr>
<tr>
<td>10th</td>
<td>-1.2680</td>
<td>.11610E-01</td>
</tr>
<tr>
<td>20th</td>
<td>-.84366</td>
<td>.51309E-01</td>
</tr>
<tr>
<td>25th</td>
<td>-.68126</td>
<td>.83422E-01</td>
</tr>
<tr>
<td>30th</td>
<td>-.53522</td>
<td>.13160</td>
</tr>
<tr>
<td>40th</td>
<td>-.26384</td>
<td>.28056</td>
</tr>
<tr>
<td>Med.</td>
<td>.48192E-01</td>
<td>.43805</td>
</tr>
<tr>
<td>60th</td>
<td>.25616</td>
<td>.71992</td>
</tr>
<tr>
<td>70th</td>
<td>.51574</td>
<td>1.0797</td>
</tr>
<tr>
<td>75th</td>
<td>.65565</td>
<td>1.3230</td>
</tr>
<tr>
<td>80th</td>
<td>.83508</td>
<td>1.6071</td>
</tr>
<tr>
<td>90th</td>
<td>1.1866</td>
<td>2.5999</td>
</tr>
<tr>
<td>Max.</td>
<td>3.0226</td>
<td>11.662</td>
</tr>
</tbody>
</table>
```

**Partition of range Minimum to Maximum**

```
<table>
<thead>
<tr>
<th>Range of X</th>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum</td>
<td>-.14333</td>
<td>.16230E-07</td>
</tr>
<tr>
<td>1st.Qrtl</td>
<td>-.6018</td>
<td>2.9154</td>
</tr>
<tr>
<td>Midpoint</td>
<td>-.60348E-01</td>
<td>5.8308</td>
</tr>
<tr>
<td>3rd.Qrtl</td>
<td>1.4811</td>
<td>8.7462</td>
</tr>
<tr>
<td>Maximum</td>
<td>3.0226</td>
<td>11.662</td>
</tr>
</tbody>
</table>
```
13.3 Histograms

The command for computing and plotting a histogram for a variable is

```
HISTOGRAM ; Rhs = the variable $
```

*LIMDEP* computes two types of histograms, for discrete (count) and for continuous data. The default type is for a frequency count of discrete data. Data are assumed to be coded 0,1,…,99. Values less than zero or above 99 are treated as out of range. A count of invalid observations is given with the output of the command. Continuous variables are assigned to 40 equal width intervals over the range of the variables. The histogram is accompanied by a table listing the relative frequencies and cumulated frequencies.

To illustrate the use of this feature, we use a data set that was employed in the study of health care system utilization, Regina T. Riphahn, Achim Wambach, and Andreas Million, ‘Incentive Effects in the Demand for Health Care: A Bivariate Panel Count Data Estimation,’ *Journal of Applied Econometrics*, Vol. 18, No. 4, 2003, pp. 387-405. The raw data were downloaded from the journal’s data archive website:


The data, which will be used in several applications below, are an unbalanced panel of observations on health care utilization by 7,293 individuals. The group sizes in the panel number as follows: $T_i$: 1=1525, 2=2158, 3=825, 4=926, 5=1051, 6=1000, 7=987. There are altogether 27,326 observations. Variables in the file are

- **hhninc** = household nominal monthly net income in German marks / 10000.
- **Hhkids** = children under age 16 in the household = 1, otherwise = 0,
- **educ** = years of schooling
- **married** = marital status
- **female** = 1 for female, 0 for male
- **docvis** = number of visits to the doctor
- **doctor** = number of doctor visits > 0
- **hospvis** = number of visits to the hospital

### 13.3.1 Histograms for Continuous Data

A histogram for the continuous variable *hhninc* would appear as follows:

```
HISTOGRAM ; Rhs = hhninc $
```

You can select the number of bars to plot with

```
; Int = k
```

This can produce less than satisfactory results, however. Ts.
There are other ways to examine continuous data. One way is to use **RECODE** to change your continuous variable into a discrete one. Alternatively, you may provide a set of interval limits and request a count of the observations in the intervals you define. The command would be

\[
\text{HISTOGRAM ; Rhs = … ; Limits = l0,l1,…,lk} \ $\n\]

where the limits you give are the left boundaries of the intervals. Thus, the number of limits you provide gives the number of intervals. Intervals are defined as ‘greater than or equal to lower and less than upper.’ For example, still using our income data,

\[
\text{HISTOGRAM ; Rhs = educ ; Limits = 0,.75, 2.5, 4, 6, 9, 12} \ $\n\]

defines five bins for the histogram, with the rightmost containing all values greater than or equal to 12. To request \( K \) equal length intervals in the range lower to upper, use

\[
\text{HISTOGRAM ; Rhs = variable ; Int = k ; Limits = lower,upper} \ $\n\]

Finally, you can use **HISTOGRAM** to search for the interval limits instead of the frequency counts. The command

\[
\text{HISTOGRAM ; Rhs = variable ; Bin = p} \ $\n\]

where ‘\( p \)’ is a sample fraction (proportion), will obtain the interval boundaries such that each bin contains the specified proportion of the observations.

**NOTE:** If the specified proportion does not lead to an even set of bins, then an extra, smaller bin is created if the remaining proportion is more than \( p/2 \). For example, if \( p \) is .22, there will be four bins with .22 and one at the right end with .12. But, if the extra mass is less than \( p/2 \), it is simply added to the rightmost bin, as for \( p = .16 \), for which the sixth bin will contain .2 of the observations.
13.3.2 Histograms for Discrete Data

The data are also inspected to determine the type and the correct number of bars to plot for a discrete variable. For a discrete variable, the plot can be exact. Once again, up to 99 bars may be displayed: For example, the count of doctor visits in the health care data appear as follows:

```
HISTOGRAM ; Rhs = docvis $
```

![Figure 13.2 Histogram for a Discrete Variable]

The long tail of the skewed distribution has rather distorted the figure. The options described earlier can be used to modify the figure. However, those options are assumed to be used for continuous data, which would distort the figure in another way. A more straightforward way to deal with the preceding situation is to operate on the data directly. For example,

```
HISTOGRAM ; If [docvis <= 25] ; Rhs = docvis $
```

truncates the distribution, but produces a more satisfactory picture of the frequency count.

13.4 Cross Tabulations

The command for crosstabs based on two variables is

```
CROSSTAB ; Lhs = rows variable ; Rhs = columns variable $
```

Use `CROSSTAB` to analyze a pair discrete of variables that are coded 0,1,… up to 49 (i.e., up to 2,500 possible outcomes). The table may be anywhere from 2×2 to 50×50. (Row and column
Describing Sample Data

13.5 Kernel Density Estimation

The kernel density estimator is a device used to describe the distribution of a variable nonparametrically, that is, without any assumption of the underlying distribution. The kernel density function for a single variable is computed using

\[ f(z_j) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{h} K \left( \frac{z_j - x_i}{h} \right), j = 1, \ldots, M. \]

The function is computed for a specified set of values \( z_j, j = 1, \ldots, M \). Note that each value requires a sum over the full sample of \( n \) values. The default value of \( M \) is 100. The primary component of the computation is the kernel function, \( K[.] \), a weighting function that integrates to one. Eight alternatives are provided:

1. Epanechnikov: \( K[z] = .75(1 - .2z^2) / \sqrt{5} \) if \( |z| \leq 5 \), 0 else,
2. Normal: \( K[z] = \phi(z) \) (normal density), \( -\infty < z < \infty \)
3. Logit: \( K[z] = \Lambda(z)(1-\Lambda(z)) \) (default), \( -\infty < z < \infty \)
4. Uniform: \( K[z] = .5 \) if \( |z| < 1 \), 0 else,
5. Beta: \( Z[z] = (1-z)(1+z)/24 \) if \( |z| < 1 \), 0 else,
6. Cosine: \( K[z] = 1 + \cos(2\pi z) \) if \( |z| < .5 \), 0 else,
7. Triangle: \( K[z] = 1 - |z| \), if \( |z| \leq 1 \), 0 else.
8. Parzen: \( K[z] = 4/3 - 8z^2 + 8|z|^3 \) if \( |z| \leq .5 \), 8(1-|z|)^3 else.
The other essential part of the computation is the smoothing (bandwidth) parameter, \( h \). Large values of \( h \) stabilize the function, but tend to flatten it and reduce the resolution (in the same manner as its discrete analog, the bin width in a histogram). Small values of \( h \) produce greater detail, but also cause the estimator to become less stable.

The basic command is

\[ \text{KERNEL} \quad ; \text{Rhs = the variable } \$

With no other options specified, the routine uses the logit kernel function, and uses a data driven bandwidth equal to

\[ h = 0.9Q/n^{0.2} \text{ where } Q = \min(\text{std.dev.}, \text{range}/1.5) \]

For an example, we will compute the kernel density that is a smoothed counterpart to the histogram for income distribution in Figure 13.1. The command is

\[ \text{KERNEL} \quad ; \text{Rhs = hhninc } \$

The results follow. The histogram is repeated to show the similarity. Once again, the very long tail of the distribution distorts the figure. We show below how to adjust the parameters to accommodate this problem. For the figure below, we used

\[ ; \text{Endpoints = 0,30} \]

to force the figures to have the same range of variation on the horizontal axis.

![Kernel Density Estimator for Incomes](image)

The kernel density also produces some summary statistics, as shown below for the example in Figure in 13.3.
Describing Sample Data

Kernel Density Estimator for HHNINC
Observations = 27326
Points plotted = 100
Bandwidth = 0.206384

Statistics for abscissa values:
Mean = 3.520836
Standard Deviation = 1.769083
Minimum = 0.000000
Maximum = 30.671000

Kernel Function = Logistic
Cross val. M.S.E. = 0.000000
Results matrix = KERNEL

The data used to plot the kernel estimator are also retained in a new matrix named (of course) kernel. Figure 13.4 shows the results for the preceding plot:

You may specify the kernel function to be used with

; Kernel = one of the eight types of kernels listed earlier

The bandwidth may be specified with

; Smooth = the bandwidth parameter

The default number of points specified is 100, with \( z_j \) = a partition of the range of the variable

You may specify the number of points, up to 200 with

; Pts = number of points to compute and plot
13.6 Scatter Plots and Plotting Data

This chapter will describe commands for producing high resolution graphs. This feature can be used for simple scatter diagrams, time series plots, and for plotting functions such as log likelihoods. You can print the graphics on standard printers and create plotter files for export to word processing programs such as Microsoft Word.

13.6.1 Printing and Exporting Figures

*LIMDEP* uses the standard Windows interface between input and output devices. When a plot appears in a window, you can use File:Print to send a copy to your printer. You can also save the graph as a Windows metafile (.WMF format) by using File:Save or File:Save As. For purposes of illustrating these functions, we will use Figure 13.5 which was generated by a PLOT command. The figure shows *LIMDEP*’s base format for graphics. Every graph generated is placed in its own scalable window, as shown in Figure 13.6, apart from the project, editing and output windows already open. This window will remain open until you close it. When you are finished reviewing the figure, you should close the window to avoid proliferating windows. You will be prompted to save the graph if you have not already done so.

![Figure 13.5 Time Series Plot Using the PLOT Command](image)

13.6.2 Saving a Graph as a Graphics File

You may save any figure from a graphics window to disk in the Windows metafile (.wmf) format. Use File:Save or File:Save As. This file type is transportable to many other programs, including Microsoft Word and Excel. For example, in Word, click the Insert menu and then select Picture, then From File to import your .wmf file into your Word document. The Windows .wmf format includes codes that allow you to scale the figure to whatever size you desire.

**TIP**: Using Edit/Copy in *LIMDEP* then Edit/Paste in Word or Excel will transport the graph from the screen to your document. However, the internal components of the figure are not fully formed by this method, and the quality of the figure will be inferior to what you will obtain by writing the figure to a .wmf file and importing the file into the other program.
Describing Sample Data

Figure 13.6 LIMDEP Desktop with Graphics Window

Figure 13.7 Excel Spreadsheet with LIMDEP .mf File Imported
13.6.3 The PLOT Command

The command for producing a basic scatter (XY) plot of one or more variables against another variable is

\[
\text{PLOT} \quad ; \ Lhs = \text{variable on horizontal axis} \\
; \ Rhs = \text{variables (up to five) on vertical axis} \quad $
\]

*Note the reversal of LIMDEP’s usual convention.* This command puts the Lhs variable on the horizontal axis, whereas a regression might be expected to do the reverse.

You may add a title to the figure by including

\[
; \ Title = \text{the title to be used}
\]

The title is placed at the top of the figure. The vertical axis of the plot is usually labeled with some variable name. You can override this with

\[
; \ Yaxis = \text{the label to be used, up to eight characters}
\]

This will often be useful when you plot a function or more than one variable.

13.6.4 Plotting One Variable Against Another

To produce a scatter plot of one variable \(y\) against another \(x\) variable, the PLOT command is given with only a single Rhs variable. The command would be

\[
\text{PLOT} \quad ; \ Lhs = x \ ; \ Rhs = y \quad $
\]

Figure E4.5 was produced with the commands listed.

\[
\text{CREATE} \quad ; \ g = \text{gasq/(100*pop/282429)}$
\]

\[
\text{PLOT} \quad ; \ Lhs = \text{gasq} \ ; \ Rhs = g \\
; \ Title = \text{Simple Plot of Gas against Price} \quad $
\]

---

Figure 13.8 Simple Scatter Plot
13.6.5 Plotting a Simple Linear Regression

To add a regression line to a figure, add

; Regression

to the PLOT command. By adding ; Regression to the preceding command, we obtain the plot in Figure E4.6. (You can also obtain this by selecting Display linear regression line in the Options page of the command builder.) (In previous versions of LIMDEP, the regression equation would replace the title in the figure. In this version, the title appears as a header and the regression equation is placed in the footer of the figure.)

Figure 13.9 Scatter Plot with Linear Regression

13.6.6 Time Series Plots

Time series plots, that is, plots of variables against the date can be obtained by using DATES and PERIOD to set up the dating, then omitting the ; Lhs part of the PLOT command. When you omit the ; Lhs part of the command, it is assumed that this is a time series plot, and the adjacent points are automatically connected. The figure is also automatically labeled with the dates. The ; Fill specification discussed below is not necessary. Figure E4.7 is a time series plot of the three macroeconomic price series for the data above. Note the use of the ; Grid specification to improve the readability of the figure. The command is

DATES ; 1953 $ 
PERIOD ; 1953 – 2004 $ 
PLOT ; Lhs = pn, pd, ps 
; Grid 
; Title = Time Series Plot of Price Series 
; Yaxis = Prices $
13.6.7 Plotting Several Variables Against One Variable

To plot several variables against a single one, just include more than one Rhs variable in the command. The command is

\[ \text{PLOT } \text{Lhs = variable on horizontal axis} \]
\[ \text{Rhs = up to five variables to be plotted} \]
\[ \ldots \text{other options, such as } \text{Grid and } \text{Fill} \]

(Note that the command builder dialog box allows you to specify multiple variables.) A different line style is used for each variable if you use \text{Fill}. (The time series plot above is an example in which the Lhs variable is the automatically supplied date.) A different type of point is constructed for each if you are using a cross section. Generally, \text{PLOT} with \text{Fill} (see the next section) creates a figure with one or more line plots, joining segments at the points, but suppressing any symbols for the points. The symbols (dots, stars, etc.) may be retained with \text{Symbols}. The \text{Regression} command is ignored if more than one variable is being plotted. An example is shown in Figure 13.11:

\[ \text{PLOT } \text{Rhs = pn, pd, ps} \]
\[ \text{Lhs = year} \]
\[ \text{Title = Scatter Plot of Price Series} \]
\[ \text{Yaxis = Prices} \]
\[ \text{Grid} \]
\[ \text{Fill} \]
\[ \text{Symbols} \]
NOTE: This is not a time series plot, in spite of the fact that year is the variable on the horizontal axis. Although at this point, LIMDEP does know that these are time series data, it does not know that ‘year’ is a date variable; year is just another variable in the data set. If you omit the ; Lhs = variable specification in the command, LIMDEP will label the x axis ‘YEAR,’ but this is not with respect to a variable in your data set; it is the date labeling that you gave in your DATES command. Even if you did not have a variable named year in your data set, you can obtain a time series style plot with yearly observations, and labeled as such.

13.6.8 Options for Scaling and Labeling the Figure

Scaling

The limits for the vertical and horizontal axes are chosen automatically so that every point appears in the figure. Boundaries are set by the ranges of the variables. You can override these settings, however. The options are as follows:

To set the limits for the horizontal axis use  ; Endpoints = lower value, upper value
To set specific limits for the vertical axis use  ; Limits = lower value, upper value

HINT: If you plot variables of very different magnitudes in the same figure, or if your series has outliers in it, the scaling convention that seeks to include every point in the graph may severely distort your figure. Missing values will also severely distort your scatter plot.

NOTE: If the endpoints or limits that you specify push any points out of the figure – x or y values are outside the limits – then the specifications are ignored, and the original default values are used.
For example, the top panel in Figure 13.12 is the same as Figure 13.9, produced by the command below without the specification of the endpoints and limits. The lower panel shows the effect of expanding the limits.

```
PLOT ; Rhs  =  gasp
; Lhs  =  g
; Title  =  Gasoline Consumption vs. Price
; Yaxis = Gas_Cons
; Grid
; Limits = 0,125  Set the vertical axis limits
; Endpoints = 0,1.2 $ Set the horizontal axis limits
```
The following describe some devices for changing the appearance of the figure, and creating particular types of graphs. Some of these have been used in the examples above. More extensive applications appear below.

**Grids and Lines in the Plotting Field**

It is sometimes helpful when plotting to put a grid in the figure. This makes it easier to relate the points in the graph to the distances on the axes. You may request a grid to be placed in the figure with :

; Grid

This divides the screen into a grid of rectangles using dotted bars. The option was used in the preceding examples. You may also put horizontal and/or vertical lines in the figure at specific numerical benchmarks. The syntax is :

; Spikes = up to five value(s) to put vertical lines at particular values
; Bars = up to five value(s) to put horizontal lines at particular values

The vertical or horizontal line is drawn from axis to axis, the full width or height of the box. The examples below use these devices to create different types of graphs.

**Connecting Points in the Plotting Field**

If you are plotting a function or a time series, it may also be useful to connect adjacent points. To do so, add :

; Fill

to the command. One way you might use this device would be to draw a function by creating a set of equally spaced values, then plotting the function of these values, connecting the points to create the continuous function.
This page intentionally left blank.
Chapter 14

The Linear Regression Model

14.1 Introduction

This chapter will detail estimation of the single equation, linear regression model

\[ y_i = x_i \beta_1 + x_2 \beta_2 + \ldots + x_k \beta_k + \epsilon_i \]

\[ = x_i' \beta + \epsilon_i, \quad i = 1, \ldots, n. \]

The full set of observations is denoted for present purposes as

\[ y = X \beta + \epsilon. \]

The initial stochastic assumptions are the most restrictive for the linear model:

\[ E[\epsilon_i | X] = 0 = E[\epsilon_i] \forall i \quad \text{(zero mean)} \]

\[ \text{Var}[\epsilon_i | X] = \text{Var}[\epsilon_i] = \sigma^2 \forall i \quad \text{(homoscedastic)} \]

\[ \text{Cov}[\epsilon_i, \epsilon_j | X] = \text{Cov}[\epsilon_i, \epsilon_j] = 0 \forall i, j \quad \text{(nonautocorrelation).} \]

14.2 Least Squares Regression

The basic command for estimating a linear regression model with least squares is

```
REGRESS ; Lhs = dependent variable
; Rhs = regressors $
```

The Rhs list may also include lagged variables and logs of variables. This requests a linear ordinary least squares regression of the Lhs variable on the set of Rhs variables. The standard output from the procedure is listed in the next section.

**NOTE:** Remember that *LIMDEP* does not automatically include a constant term in the equation. If you want one, be sure to include *one* among the Rhs variables.

An example of the standard output for a linear regression appears below.
The Linear Regression Model

Ordinary least squares regression
Model was estimated Dec 13, 2007 at 08:34:51AM
LHS=LOGG Mean = -.2571288
Standard deviation = .2384918
Number of observs. = 52
Model size Parameters = 6
Degrees of freedom = 46
Residuals Sum of squares = .1070036
Standard error of e = .4823033E-01
Fit R-squared = .9631123
Adjusted R-squared = .9591028
Model test F[ 5, 46] (prob) = 240.21 (.0000)
Diagnostic Log likelihood = 87.05475
Restricted(b=0) = 1.257919
Chi-sq [ 5] (prob) = 171.59 (.0000)
Info criter. LogAmemiya Prd. Crt. = -5.954335
Akaike Info. Criter. = -5.955367
Autocorrel Durbin-Watson Stat. = .2512498
Rho = cor[e,e(-1)] = .8743751

| Variable | Coefficient  | Standard Error | t-ratio | P[|T|>t] | Mean of X |
|----------|--------------|----------------|--------|---------|----------|
| Constant | -11.5997167  | 1.48817387     | -7.795 | .0000   |          |
| LOGPG    | -.03438256   | .04201503      | -.818  | .4174   | 3.72930296|
| LOGINC   | 1.31596815   | .14198287      | 9.268  | .0000   | 9.67487347|
| LOGPNC   | -.11963708   | .20384305      | -.587  | .5601   | 4.38036655|
| LOGPUC   | .03754405    | .09814077      | .383   | .7038   | 4.10544880|
| LOGPPT   | -.21513953   | .11655849      | -1.846 | .0714   | 4.14194132|

The statistics reported are as follows:

- The model framework – linear least squares regression
- The date and time when the estimates were computed
- Name of the dependent variable
- Mean of Lhs variable \( \bar{y} = \frac{1}{n} \sum y_i \)
- Standard deviation of Lhs variable \( \left\{ \frac{1}{(n-1)} \{ \sum_{i=1}^{n} (y_i - \bar{y})^2 \} \right\}^{1/2} \)
- Name of the weighting variable if one was specified
- Number of observations \( n \)
- Number of parameters in regression \( K \)
- Degrees of freedom \( n-K \)
- Sum of squared residuals \( e'e = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} (y_i - x'b)^2 \)
- Standard error of e \( s = \sqrt{e'e/(n-K)} \)
- \( R^2 \) \( R^2 = 1 - \frac{e'e}{\sum_{i=1}^{n} (y_i - x'b)^2} \)
- Adjusted \( R^2 \) \( \bar{R}^2 = 1 - (n-1)/(n-K)[1 - R^2] \)
- F statistic \( F[K-1,n-K] = \left[ R^2/(K-1) \right] / \left[ (1-R^2)/(n-K) \right] \)
- Prob value for F \( Prob_F = \text{Prob}[F[K-1,n-k] > \text{observed } F] \)
- Log likelihood \( \log L = -n/2[1 + \log 2\pi + \log(e'e/n)] \)
- Restricted log likelihood \( \log L_0 = -n/2[1 + \log 2\pi + \log((1/n)\sum_{i=1}^{n} (y_i - \bar{y})^2)] \)
- Chi squared\([K-1]\) \( \chi^2 = 2(\log L - \log L_0) \)
- Prob value for chi squared \( Prob_{\chi^2} = \text{Prob}[\chi^2(K-1)] > \text{observed chi squared} \)
- Log Amemiya Prediction Criterion \( = \log[s^2 (1 + K/n)] \)
The Linear Regression Model

- Akaike Information Criterion $AIC = (\log L - K)/(n/2) - (1 + \log 2\pi)$
- Durbin-Watson $dw = \frac{\sum_{t=2}^{T} (e_t - e_{t-1})^2}{\sum_{t=1}^{T} e_t^2}$
- Autocorrelation $r = 1 - dw/2$

The $R^2$ and related statistics are problematic if your regression does not contain a constant term. For the linear model, LIMDEP will check your specification, and issue a warning in the output. Finally, the main table of regression results contains, for each Rhs variable in the regression:

- Name of the variable,
- Coefficient $b_k$,
- Standard error of coefficient estimate $se_k = \{[s^2(XX)^{-1}]_{kk}\}^{1/2}$.
- $t$ ratio for the coefficient estimate $t_k = b_k / se_k$
- Significance level of each $t$ ratio based on the $t$ distribution with $[n-K]$ degrees of freedom $= p$ value $= \text{Prob}[t(n-K)] > \text{observed } t_k$
- Sample mean of the variable.

14.2.1 Retrievable Results

The retrievable results which are saved automatically by the REGRESS command are

Matrices:
- $b$ = slope vector $= (X'X)^{-1} X'y$
- $varb$ = estimated covariance matrix $= [e'e/(n-K)](X'X)^{-1}$

Scalars:
- $ssqrd = e'e/(n-K)$
- $rsqrd = R^2$
- $s = s$
- $sumsqdev = \text{sum of squared deviations, } e'e$
- $rho = \text{autocorrelation coefficient, } r$
- $degfrdm = n-K$
- $sy = \text{sample standard deviation of Lhs variable}$
- $ybar = \text{sample mean of Lhs variable}$
- $kreg = \text{number of independent variables, } K$
- $nreg = \text{number of observations used to compute the regression, } n$

Note, this may differ from the sample size if you have skipped observations containing missing values.
- $logl = \text{log likelihood}$
- $exitcode = 0.0$ unless the data were collinear or OLS gives a perfect fit

Last Model: $b_{\text{name}}$ where the names are the Rhs variables.
(See WALD in Section R11.5.2)

14.2.2 Predictions and Residuals

To obtain a list of the residuals and fitted values from a linear regression model, add the specification

; List

to the command. The residuals and predicted values may be kept in your data area by using the specifications

; Res = name for residuals
The Linear Regression Model

and

; Keep = name for predicted values

If you are not using the full sample or all of the rows of your data matrix, some of the cells in these columns will be marked as missing. If you have data on the regressors but not the dependent variable, you can use

; Fill

to obtain predictions for the missing data. Remember, though, that the prediction is -999 (missing) for any observation for which any of the xs are missing.

The following commands could be used for computing forecast standard errors. This routine uses the matrices \( b \) (the coefficients) and \( varb \) (estimated covariance matrix) kept by the regression and scalar \( ssqrd \) which is s-squared from the regression. The forecast standard errors are the values computed inside the Sqr function in the CREATE command.

```plaintext
NAMELIST ; x = the set of regressors  $
REGRESS  ; Lhs = y ; Rhs = x ; Keep = yhat  $
CALC  ; ct = Ttb(.975,degfrdm)  $
CREATE  ; lowerbnd  = yhat -  ct * Sqr(ssqrd + Qfr(x,varb))
; upperbnd  = yhat + ct * Sqr(ssqrd + Qfr(x,varb))  $
```

A plot of the residuals from your regression can be requested by adding

; Plot

to the command. Residuals are plotted against observation number (i.e., simply listed). If you would like to plot them against another variable, change the preceding to

; Plot(variable name)

These variables can be any existing variables. They need not have been used in the regression. The residuals are sorted according to the variable you name and plotted against it. The plot will show the residuals graphed against either the observation number, the date for time series data, or the variable you specify using the ; Plot(variable) option described above.

- If there are outliers in the data, this may severely cramp the figure, since the vertical axis is scaled so that every observation will appear.

- The mean residual bar may not appear at zero because the residuals may not have zero mean. They will not if you do not have a constant term in your regression or if you are plotting two stage least squares residuals. Since 0.0 will generally not be the midpoint between the high and low residual, the zero bar will not be in the center of your screen even when you do have a constant term in the model.
14.2.3 Robust Covariance Matrix Estimation

**REGRESS** will compute robust estimators for the covariance matrix of the least squares estimator for both heteroscedastic and autocorrelated disturbances. Although OLS is generally quite robust, some researchers have advocated other estimators for finite sample purposes. **REGRESS** can also be used to compute the least absolute deviations estimator.

**Heteroscedasticity – The White Estimator**

For the heteroscedasticity corrected (White) estimator, use

; Heteroscedasticity

in the **REGRESS** command. The White estimator is

\[
\text{Est. Var}[\beta] = (X'X)^{-1} \times \sum_{i=1}^{n} e_i^2 x_i x'_i \times (X'X)^{-1}
\]

**Autocorrelation – The Newey-West Estimator**

The Newey-West robust estimator for the covariance matrix of the least squares estimator in the presence of autocorrelation is

\[
\text{Est. Var}[\beta] = (X'X)^{-1} \times \sum_{t=1}^{T} e_t^2 x_t x'_t \times (X'X)^{-1}
\]

\[
+ (X'X)^{-1} \times \left\{ \frac{1}{T} \sum_{j=1}^{L} \sum_{t=j+1}^{T} \left( 1 - \frac{j}{L+1} \right) e_t e_{t-j} [x_t x_{t-j}' + x_{t-j} x'_t] \right\} \times (X'X)^{-1}
\]

You (the analyst) must provide the value of \(L\), the number of lags for which the estimator is computed. Then, request this estimator by adding

; Pds = ... the value for \(L\)

to the **REGRESS** command.

14.2.4 Restricted Least Squares

This section describes procedures for estimating the restricted regression model,

\[
y = X\beta + e
\]

subject to \(R\beta = q\).

\(R\) is a \(J \times K\) matrix assumed to be of full row rank. That is, we impose \(J\) linearly independent restrictions. They may be equality restrictions, inequality restrictions, or a mix of the two. Inequality restricted least squares is handled by a quadratic programming method that is documented in the full manual.
The constrained ordinary least squares estimator is

$$b_c = b - (X'X)^{-1}R[(RX')^{-1}R']^{-1}(Rb-q)$$

where

$$b = (X'X)^{-1}X'y$$

is the unrestricted least squares estimator. The estimator of the variance of the constrained estimator is

$$\text{Est. Var}[b_c] = s^2(X'X)^{-1} - s^2(X'X)^{-1}R[(RX')^{-1}R']^{-1}R(X'X)^{-1}.$$ 

where

$$s^2 = (y - X b_c)'(y - X b_c)/(n-K+J).$$

The parameter vector is written $b = b_1, b_2, ..., b_K$ where the correspondence is to your Rhs list of variables, including the constant, one if it is included. The restrictions are then imposed algebraically with

$$; \text{CLS: linear function} = \text{value},$$
$$\quad \text{linear function} = \text{value}, ...$$

For example, the following imposes constant returns to scale (capital coefficient + labor coefficient = 1) on a (hypothetical) production function:

```
CALC ; Ran (123457) $
SAMPLE ; 1 – 100 $
CREATE ; l  = Rnu(1,3)
; k = Rnu(.5,2)
; y = Exp(3 + .6 * Log(l) + .4 * Log(k) + Rnn(0,4)) $
REGRESS ; Lhs = Log(y) ; Rhs = one,log(l),log(k)
; CLS: b(2) + b(3) = 1 $
```

The following results for restricted least squares are produced. (The CALC function sets the seed for the random number generator at a specific value, so you can replicate the results.)

<table>
<thead>
<tr>
<th>Ordinary least squares regression</th>
</tr>
</thead>
<tbody>
<tr>
<td>LHS=LOGY</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>WTS=none</td>
</tr>
<tr>
<td>Model size</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Residuals</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Fit</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Model test</td>
</tr>
<tr>
<td>Diagnostic</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

| Variable | Coefficient | Standard Error | t-ratio | P[|T|>t] | Mean of X |
|----------|-------------|----------------|---------|---------|-----------|
| Constant | 3.20014797 | 1.06375203 | 3.008 | .0033 |
| LOGL | .80563172 | 1.40618551 | .573 | .5680 | .68423500 |
| LOGK | -.85445443 | 1.14222281 | -.748 | .4562 | .15523890 |
The Linear Regression Model

<table>
<thead>
<tr>
<th>Linearly restricted regression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ordinary least squares regression</td>
</tr>
<tr>
<td>LHS = LOGY</td>
</tr>
<tr>
<td>Standard deviation = 4.069505</td>
</tr>
<tr>
<td>WTS = none</td>
</tr>
<tr>
<td>Model size</td>
</tr>
<tr>
<td>Degrees of freedom = 98</td>
</tr>
<tr>
<td>Residuals</td>
</tr>
<tr>
<td>Standard error of e = 4.078146</td>
</tr>
<tr>
<td>Fit</td>
</tr>
<tr>
<td>Adjusted R-squared = -0.4251326E-02</td>
</tr>
<tr>
<td>Diagnostic</td>
</tr>
<tr>
<td>Info criter. LogAmemiya Prd. Crt. = 2.831088</td>
</tr>
<tr>
<td>Akaike Info. Criter. = 2.831082</td>
</tr>
<tr>
<td>Autocorrel Durbin-Watson Stat. = 2.064377</td>
</tr>
<tr>
<td>Rho = cor[e,e(-1)] = -0.321885</td>
</tr>
<tr>
<td>Restrictns. F[ 1, 97] (prob) = 0.33 (0.5678)</td>
</tr>
<tr>
<td>Not using OLS or no constant. Rsqd &amp; F may be &lt; 0.</td>
</tr>
<tr>
<td>Note, with restrictions imposed, Rsqd may be &lt; 0.</td>
</tr>
</tbody>
</table>

You may impose as many restrictions as you wish with this estimator; simply separate the restrictions with commas.

14.2.5 Hypothesis Tests in the Linear Model

The REGRESS and MATRIX commands can be used to test a variety of hypotheses.

The $F$ statistic for testing the set of $J$ restrictions

$$H_0: R\beta = q$$

is

$$F[J,n-K] = \frac{(Rb - q)'[s^2R(X'X)^{-1}R']^2(Rb - q)}{J}$$

$$= \frac{[(e'^e - e'^e)/J]}{[e'^e/(n-K)]}$$

where the subscript ‘*’ indicates the sum of squares with the restrictions imposed and $b$ is the unrestricted ordinary least squares estimator. This statistic is included in the diagnostic table whenever you use ; CLS: to impose linear restrictions.

Consider an example, where now we make two restrictions hold as equalities:

SAMPLE  ; 1 - 500 $\$
CALC  ; Ran(123457) $\$
CREATE  ; l = Rnu(1,3) ; k = Rnu(.5,2) ; ll = Log(l) ; lk = Log(k)
; lk2 = lk*lk ; ll2 = ll*ll ; lkl = ll*lk
; ly = 3 + .6*l + .4*k - .05*ll2 - .15*lk2 + .2*lkl + Rnn(0,4) $\$
REGRESS  ; Lhs = ly ; Rhs = one, ll, lk, ll2, lk2, lkl
; CLS: b(2)+b(3) = 1, b(4)+b(5)+b(6) = 0 $
| Variable | Coefficient | Standard Error | t-ratio | P[|T|>|t|] | Mean of X |
|----------|-------------|----------------|---------|------------|----------|
| Constant| 3.40835358  | .76285798      | 4.468   | .0000      |
| LL      | -0.09422513 | 2.56561134     | -0.037  | .9707      | .66349271 |
| LK      | -0.05073476 | 1.16722384     | -0.043  | .9653      | .16198512 |
| LL2     | 0.54115572  | 2.06524209     | 0.262   | .7934      | .53150388 |
| LK2     | -0.70218471 | 1.38385662     | -0.507  | .6121      | .16242739 |
| LKL     | -0.15810258 | 1.55538131     | -0.102  | .9191      | .10542795 |

---

| Variable | Coefficient | Standard Error | t-ratio | P[|T|>|t|] | Mean of X |
|----------|-------------|----------------|---------|------------|----------|
| Constant| 2.87115229  | .26637740      | 10.779  | .0000      |
| LL      | 1.16811718  | .94623345      | 1.234   | .2176      | .66349271 |
| LK      | -0.16811718 | .94623345      | -0.178  | .8591      | .16198512 |
| LL2     | -3.3357188  | 1.05355989     | -3.317  | .7517      | .53150388 |
| LK2     | 0.31357011  | 0.76300427     | 0.411   | .6813      | 0.1624739 |
| LKL     | 0.02000176  | 1.26678767     | 0.016   | 0.9874     | 0.10542795 |
14.3 Estimating Models with Heteroscedasticity

For the model in which \( \omega_i \) is either known or has been estimated already, the weighted least squares estimator is requested with

```
REGRESS ; Lhs = ... ; Rhs = ... ; Wts = weighting variable $
```

In computing weighted estimators, we use the formulas:

\[
\begin{align*}
  n & = \text{the current sample size, after skipping any missing observations,} \\
  w_i & = \left( \frac{n}{\sum_i W_i} \right) W_i = \text{Scale} \times W_i \quad \text{(note that} \sum_i w_i = n), \\
  b_w & = \left[ \sum_i w_i x_i x_i' \right]^{-1} \left[ \sum_i w_i x_i y_i \right], \\
  s_w^2 & = \sum_i w_i (y_i - x_i' b_w)^2, \\
  \text{Est.Var.}[b_w] & = \left[ s_w^2 / (n-K) \right] \left[ \sum_i w_i x_i x_i' \right]^{-1},
\end{align*}
\]

where \( W_i \) is your weighting variable. Your original weighting variable is not modified (scaled) during this computation. The scale factor is computed separately and carried through the computations.

**NOTE:** Apart from the scaling, your weighting variable is the reciprocal of the *individual specific variance*, not the standard deviation, and not the reciprocal of the standard deviation. This construction is used to maintain consistency with the other models in LIMDEP.

For example, consider the common case, \( \text{Var}[\varepsilon_i] = \sigma^2 z_i^2 \). For this case, you would use

```
CREATE ; wt = 1 / z ^ 2 $
REGRESS ; Lhs = ... ; Rhs = ... ; Wts = wt $
```

14.4 Correcting for First Order Autocorrelation

There are numerous procedures for estimating a linear regression with first order autoregressive disturbances,

\[
\begin{align*}
  y_i & = \beta' x_i + \varepsilon_i, \\
  \varepsilon_i & = \rho \varepsilon_{i-1} + u_i.
\end{align*}
\]

The simplest form of the command is

```
REGRESS ; Lhs = ... ; Rhs = ... ; AR1
```

The default estimator is the iterative Prais-Winsten algorithm. That is, the first observation is *not* discarded; the full GLS transformation is used. This is a repeated two step estimator:

**Step 1.** OLS regression of \( y \) on \( X \). Then, estimate \( \rho \) with \( r = 1 - 0.5 \times \) Durbin-Watson statistic.

**Step 2.** OLS regression of \( y_t^* = (1 - r)^{1/2} y_t \) and \( y_t^* = y_t - r y_{t-1}, t = 2, ..., T \) on the same transformation of \( x_t \).
After Step 2, \( r \) is recomputed based on the GLS estimator, and the regression is repeated. This iteration continues until the change in \( r \) from one iteration to the next is less than 0.0001. The covariance matrix for the slope estimators is the usual OLS estimator, \( s^2(X^*X^*)^{-1} \) based on the transformed data. The asymptotic variance for \( r \) is estimated by \( (1 - r^2)/(T-1) \). Results and diagnostics are presented for both transformed and untransformed models.

**NOTE:** If no other specification is given, the estimator is allowed to iterate to convergence, which usually occurs after a small number of iterations. The updated value of \( r \) at each iteration is computed from the Durbin-Watson statistic based on the most recent GLS coefficients estimates. Iterating these estimators to convergence does not produce a maximum likelihood estimator.

Other estimation procedures are requested by adding them to the ; `AR1` request:

; `AR1 ; Alg = Corc`

requests the iterative Cochrane-Orcutt estimator. The first observation is skipped, and the pseudo-difference defined above is applied to the remaining observations. (We do not recommend this estimator, as it needlessly discards the information contained in the first observation, with no accompanying gain in speed, efficiency, or any statistical properties.) Alternatively,

; `AR1 ; Alg = MLE`

requests the maximum likelihood estimator of Beach and MacKinnon (1978). In this model, the MLE is not GLS because in addition to the generalized sum of squares, the log likelihood function contains an extra term, the Jacobian for the first observation, \( \frac{1}{2} \log(1 - \rho^2) \). This term becomes deminimus as \( T \to \infty \), so in a large sample, the MLE and the other GLS estimators should not differ substantially.

To use a grid search for the autocorrelation coefficient, use

; `AR1 ; Alg = grid(lower, upper, step)`

This requests a simple grid search over the indicated range with a stepsize as given. The method used for the grid search is the default Prais-Winsten estimator. To request the Cochrane-Orcutt estimator, instead, use

; `AR1 ; Alg = grid(lower, upper, step, 1)`

(As before, the Cochrane-Orcutt estimator is inferior to the MLE or Prais-Winsten estimator.) You can request a particular value for \( \rho \) by a simple request:

; `AR1 ; Rho = specific value`

When you use this form of the model command, the output will still contain an estimated standard error for the estimate of \( \rho \), as if it had been estimated. The number of iterations allowed for the first three estimators can be controlled with the specification

; `Maxit = maximum`
14.5 Two Stage Least Squares

The essential command fitting linear models by instrumental variables is

\[
\text{2SLS} \quad ; \ Lhs = \text{dependent variable} \\
; \ Rhs = \text{list of right hand side variables (all)} \\
; \ Inst = \text{list of all instrumental variables, including one } \\
\]

The command for computing instrumental variables or two stage least squares estimates differs from that for ordinary least squares (\texttt{REGRESS}) only in a list of instrumental variables. All options are the same as for the linear regression model – see Chapter E5 for details. This includes the specifications of \texttt{AR1} disturbances, \texttt{Plot} for residuals, etc. The list of instruments may include any variables existing in the data set.

\textbf{HINT:} If your equation (Rhs) includes a constant term, \textit{one}, then you should also include \textit{one} in the list of instrumental variables. Indeed, it will often be the case that Inst should include \textit{one} even if the Rhs does not.

Computations use the standard results for two stage least squares. (See, e.g., Greene (2008) There are no degrees of freedom corrections for variance estimators when this estimator is used. All results are asymptotic, and degrees of freedom corrections do not produce unbiased estimators in this context. Thus,

\[
\hat{\sigma}^2 = \frac{1}{n} \sum_i (y_i - \hat{\beta}'x_i)^2.
\]

This is consistent with most published sources, but (curiously enough) inconsistent with most other commercially available computer programs. It will show up as a proportional difference in all estimated standard errors. If you would prefer that the degrees of freedom correction be made, add the specification

; Dfc

to your \texttt{2SLS} command.

14.5.1 Robust Estimation of the 2SLS Covariance Matrix

The White and Newey-West robust estimators of the covariance matrix of the least squares estimator described in Section 14.2.3 can also be obtained for 2SLS by requesting it in the same fashion. All necessary corrections for the use of the instrumental variables are made in the computation.

14.5.2 Model Output for the 2SLS Command

The output for the \texttt{2SLS} command is identical to that for \texttt{REGRESS}. The only indication that 2SLS, rather than OLS, was used in estimating the model will be a line at the top of the model results indicating that two stage least squares was used in the computations and a listing of the instrumental variables that will appear above the coefficient estimates. All retrievable results and methods for testing hypotheses are likewise identical.
14.6 Panel Data Models

This chapter will detail estimation of linear models for panel data. The essential structure for most of them is an ‘effects’ model,

\[ y_{it} = \alpha_i + \gamma_t + \beta'x_{it} + \varepsilon_{it} \]

in which variation across groups (individuals) or time is captured in simple shifts of the regression function – i.e., changes in the intercepts. These models are the fixed effects (FE) and random effects (RE) models.

The commands for estimation of these models are variants of the basic structure

```
REGRESS ; Lhs = y ; Rhs = x...
; Panel
; Str = the name of a stratification variable
or
; Pds = specification of the number of periods, variable or fixed
; ... other options $
```

14.6.1 Data Arrangement and Setup

Your data are assumed to consist of variables:

\[ y_{it}, x_{1it}, x_{2it}, \ldots, x_{Kit}, I_i, \quad i = 1,\ldots,N, \quad t = 1,\ldots,T_i, \]

- \( y_{it} \) = dependent variable,
- \( x_{it} \) = set of independent variables,
- \( I_{it} \) = stratification indicator,
- \( K \) = number of regressors, not including one,
- \( N \) = number of groups,
- \( T_i \) = number of observations in group ‘i.’

The data set for all panel data models will normally consist of multiple observations, denoted \( t = 1,\ldots,T_i \), on each of \( i = 1,\ldots,N \) observation units, or ‘groups.’ A typical data set would include observations on several persons or countries each observed at several points in time, \( T_i \), for each individual. In the following, we use ‘t’ to symbolize ‘time’ purely for convenience. The panel could consist of \( N \) cross sections observed at different locations or \( N \) time series drawn at different times, or, most commonly, a cross section of \( N \) time series, each of length \( T_i \). The estimation routines are structured to accommodate large values of \( N \), such as in the national longitudinal data sets, with \( T_i \) being as large or small as dictated by the study but not directly relevant to the internal capacity of the estimator.

We define a balanced panel to be one in which \( T_i \) is the same for all \( i \), and, correspondingly, an unbalanced panel is one in which the group sizes may be different across \( i \).

**NOTE:** Panels are never required to be ‘balanced.’ That is, the number of time observations, \( T_i \), may vary with ‘i.’ The computation of the panel data estimators is neither simpler nor harder with constant \( T_i \). No distinction is made internally. There are some theoretical complications, though.
Data Arrangement

Data for the panel data estimators in LIMDEP are assumed to be arranged contiguously in the data set. Logically, you will have

\[ N_{obs} = \sum_{i=1}^{N} T_i \]

observations on your independent variables, arranged in a data matrix

\[
X = \begin{bmatrix}
T_1 \text{ observations for group 1} \\
T_2 \text{ observations for group 2} \\
\vdots \\
T_N \text{ observations for group } N
\end{bmatrix}
\]

and likewise for the data on \( y \), the dependent variable. When you first read the data into your program, you should treat them as a cross section with \( N_{obs} \) observations. The partitioning of the data for panel data estimators is done at estimation time.

**NOTE**: Missing data are handled automatically by this estimator. You need not make any changes in the current sample to accommodate missing values – they will be bypassed automatically. Group sizes and all computations are obtained using only the complete observations. Whether or not you have used **SKIP** to manage missing values, this estimator will correctly arrange the complete and incomplete observations.

Specifying the Stratification for Balanced Panels

If you have a balanced panel, that is the same number of observations for each group, you can use

\[
; \text{Pds} = T
\]

(where you give the specific value of \( T \)) to define the panel. There is no need for a stratification variable. For example, in one of our applications, we use a balanced panel data set with \( N = 10 \) firms and \( T = 20 \) years. The regression command is

\[
\text{REGRESS ; Lhs = i ; Rhs = one,f,c ; Panel ; Pds = 20 $}
\]

Specifying the Stratification for Unbalanced Panels

If you will be using unbalanced panels, you may use a stratification variable. The variable is given in the

\[
; \text{Str} = \text{group}
\]

part of the command. The stratification variable, \( \text{group} \), may be any indicator that distinguishes the groups – that is, a firm number, any kind of identification number, a telephone number, etc.
For example, the following could be used for our 100 observation panel mentioned earlier

```
REGRESS ; Lhs = i  
   ; Rhs = one,f,c  
   ; Str = firm  
   ; Panel $  
```

**Specifying the Stratification with a Count Variable**

This and all other panel data estimators in *LIMDEP* use a count variable to specify the stratification in a panel. Your panel is specified with

```
; Pds = a specific number of periods  
```

as discussed above, or with

```
; Pds = a variable which gives the group counts  
```

For an example of such a variable, suppose a panel consists of three, then four, then two observations. The nine values taken by the variable, say \( n_i \), would be 3,3,3,4,4,4,4,2,2. The panel specification would be

```
; Pds = n_i  
```

You may also use this kind of stratification indicator with the linear regression models discussed here.

**The Stratification Variable**

*LIMDEP* inspects your stratification variable before estimation of these models begins. If you have specified a balanced panel with ; Pds = T or if your stratification variable is not a consecutive sequence of integers 1,2,...,\( N \), then a new stratification variable called \( _{stratum} \) is automatically created for you. The new variable will consist of the consecutive integers. Thereafter, for the same panel, you can use this created variable for your indicator. Suppose, for example, you have a panel data set indicated by postal codes, which are not sorted in your data set (though we assume that all observations for a particular postal code appear together). Then,

```
REGRESS ; ... ; Str = postcode $  
```

would automatically create the new stratification variable. Obviously, it is redundant for current purposes, but you might have use for it later. Looking ahead, suppose you were going to fit a linear regression model to one variable in your panel, and a probit model to a different one. For the first one, you can use your postal code to specify the panel. But, the probit model needs a count variable. The next section shows how to obtain this variable.

**The Count Variable**

In addition to the stratification variable, \( _{stratum} \), this estimator also always creates a count variable named \( _{group}ti \). If you have a balanced panel, the count variable will just equal the
number of periods. But, if your panel is unbalanced, \_groupi is exactly what you will need for the other panel data estimators in \textit{LIMDEP}.

### Using REGRESS to Create Stratification and Count Variables

The preceding suggests an extremely helpful tool that should be useful elsewhere in \textit{LIMDEP}, so much so that a special form has been provided for you to use. If you have a stratification variable that identifies the group that an individual is in, you will need a count variable constructed from this in order to use the other panel data estimators in \textit{LIMDEP}. The preceding suggests that computing a panel data based linear regression might be a good way to create the variable, so a method is set up whereby you can use \textbf{REGRESS} without actually computing the regression (which you might not have any interest in) to compute this variable. The method is to use the following command:

\begin{verbatim}
REGRESS  ; Lhs = one  ; Rhs = one  \\
; Str = the stratification variable ; Panel $
\end{verbatim}

Notice that this attempts to regress \textit{one} on \textit{one}, which is not a regression at all. \textit{LIMDEP} will notice this, and respond to such a command with results such as the following:

\begin{verbatim}
| No variables specified. Stratification and count |  \\
| variables were created. Ngroup = 6 |  \\
| Full sample with missing data = 20 |  \\
| Stratification variable is _STRATUM |  \\
| Count variable for N(i) is _GROUPTI |  \\
\end{verbatim}

A variable \textit{id} with the results of the command are shown in Figure 14.1.

### 14.6.2 One Way Fixed and Random Effects Models

The next several sections consider formulation and estimation of one way common effects models,

\[ y_{it} = \alpha_i + \beta'x_{it} + \varepsilon_{it}. \]

The fixed effects model is

\[ y_{it} = \alpha_1 d_{1i} + \alpha_2 d_{2i} + \ldots + \beta'x_{it} + \varepsilon_{it} \]

\[ = \alpha_i + \beta'x_{it} + \varepsilon_{its}, \]

where

\[ E[\varepsilon_{it}|X_i] = 0, \ Var[\varepsilon_{it}|X_i] = \sigma^2, \ Cov[\varepsilon_{it},\varepsilon_{js}|X_i,X_j] = 0 \text{ for all } i,j, \]

\[ Cov[\alpha_i,x_{it}] \neq 0. \]

The efficient estimator for this model in the base case is least squares. The random effects model is

\[ y_{it} = \alpha + \beta'x_{it} + \varepsilon_{it} + u_i \]

where

\[ E[u_i] = 0, \ Var[u_i] = \sigma_u^2 \]

\[ \text{Cov}[\varepsilon_{it},u_i] = 0. \]

\[ \text{Var}[\varepsilon_{it} + u_i] = \sigma^2 = \sigma_{\nu}^2 + \sigma_u^2. \]
For a given $i$, the disturbances in different periods are correlated because of their common component, $u_i$,
\[
\text{Corr}[\varepsilon_{it} + u_i, \varepsilon_{is} + u_i] = \rho = \frac{\sigma_u^2}{\sigma^2}.
\]

The efficient estimator is generalized least squares. Before considering the various different specifications of the two models, we describe two standard devices for distinguishing which seems to be the more appropriate model for a given data set.

**Specification Tests for the One Factor Models**

Breusch and Pagan’s Lagrange multiplier statistic,
\[
\text{LM} = NT \left[ \frac{ \sum_i \left( \sum_t e_{it}^2 \right)^2 }{ \sum_i \sum_t e_{it}^2 - 1 } \right]^{\frac{1}{2}}
\]

is used to test the null hypothesis that there are no group effects in the random effects model. Arguably, a rejection of the null hypothesis is as likely to be due to the presence of fixed effects. The statistic is computed from the ordinary least squares residuals from a pooled regression. Large values of LM favor the effects model over the classical model with no common effects.
A second statistic is Hausman’s chi squared statistic for testing whether the GLS estimator is an appropriate alternative to the LSDV estimator. Computation of the Hausman statistic requires estimates of both the random and fixed effects models. The statistic is

\[ H = (\hat{\beta}_{fgls} - \hat{b}_{lsdv})' \{\text{Est.Var}[\hat{b}_{lsdv}] - \text{Est.Var}[\hat{\beta}_{fgls}]\}^{-1}(\hat{\beta}_{fgls} - \hat{b}_{lsdv}). \]

Large values of \( H \) weigh in favor of the fixed effects model. See Greene (2008) for details. Note that in some data sets, \( H \) cannot be computed because the difference of the two covariance matrices is not positive definite. (The algebra involved does not guarantee this.) Some authors suggest computing a generalized inverse in such a case, which will force the issue. But, it will not produce an appropriate test statistic. The better strategy in such a case is to take the difference between the two estimators to be random variation, which would favor the random effects estimator.

These two statistics are presented as part of the results when you fit both the fixed effects and random effects models. The \texttt{REGRESS} command can be stated to request specifically either of the two. But, if you do not specify either in particular, \texttt{LIMDEP} fits both, and near the end of the voluminous output, presents a table such as the following, which contains the LM and H statistics.

```
| Random Effects Model: v(i,t) = e(i,t) + u(i)     |
| Estimates: Var[e] = .322976D+04 |
| Var[u] = .296782D+04  |
| Corr[v(i,t),v(i,s)] = .478868   |
| Lagrange Multiplier Test vs. Model (3) = 188.08   |
|     (1 df, prob value = .000000)                   |
| (High values of LM favor FEM/REM over CR model.)  |
| Fixed vs. Random Effects (Hausman) = 3.17         |
|     (2 df, prob value = .205357)                   |
| (High (low) values of H favor FEM (REM).)         |
| Reestimated using GLS coefficients:               |
| Original: Var[e] = .141468D+05  |  |
| Var[u] = .325318D+05  |
| Sum of Squares .572940D+06 |
| R-squared  .747284D+00 |
```

\textbf{HINT:} Large values of the Hausman statistic argue in favor of the fixed effects model over the random effects model. Large values of the LM statistic argue in favor of one of the one factor models against the classical regression with no group specific effects. A large value of the LM statistic in the presence of a small Hausman statistic (as in our application) argues in favor of the random effects model.

**Commands for One Factor Common Effects Models**

The basic command for estimation of the fixed effects model is

```
REGRESS ; Lhs = dependent variable  
; Rhs = independent variables  
; Str = stratification or ; Pds = count (variable)  
; Panel ; Fixed Effects $ 
```
The command for random effects is

```
REGRESS ; Lhs = dependent variable
; Rhs = independent variables
; Str = stratification or ; Pds = count (variable)
; Panel ; Random Effects $
```

If the last line specifies only : Panel, and neither fixed nor random effects, then both models are fit. (When you fit a random effects model, the fixed effects regression may be computed in the background without reporting the results, in order to use the estimates to compute the variance components.)

### 14.6.3 One Way Fixed Effects Models

The two models estimated with this program are ‘one way’ or ‘one factor’ designs of the form

\[
y_{it} = \mu_i + \beta'x_{it} + \epsilon_{it}
\]

where \( \epsilon_{it} \) is a classical disturbance with \( E[\epsilon_{it}] = 0 \) and \( \text{Var}[\epsilon_{it}] = \sigma_{\epsilon}^2 \). In the fixed effects model, \( \mu_i \) is a separate constant term for each unit. Thus, the model may be written

\[
y_{it} = \alpha_1d_{1it} + \alpha_2d_{2it} + \ldots + \beta'x_{it} + \epsilon_{it}
\]

\[
= \alpha_i + \beta'x_{it} + \epsilon_{it},
\]

where the \( \alpha_i \)s are individual specific constants, and the \( d_s \)s are group specific dummy variables which equal one only when \( j = i \). The fixed effects model is a classical regression model. The complication for the least squares procedure is that \( N \) may be very large so that the usual formulas for computing least squares coefficients are cumbersome (or impossible) to apply. The model may be estimated in a simpler form by exploiting the algebra of least squares. This model adds the covariates \( x_{it} \) to the ‘model’ of the previous section.

### Commands for One Way Fixed Effects Models

To invoke this procedure, use the command

```
REGRESS ; Lhs = y
; Rhs = list of regressors
; Str = stratification variable or ; Pds = number of periods
; Panel
; Fixed Effects $
```

You need not include one among your regressors. The constant is placed in the regression automatically when it is needed. You may also use:

```
; Output = 2
```

to list fixed effects in an output file. This will also produce estimated standard errors for the fixed effects. If the number of groups is large, the amount of output can be very large.
The Linear Regression Model

Standard options for residuals and fitted values, include the following. All of

; List  to display residuals and fitted values
; Keep = name to retain predictions
; Res  = name to retain residuals
; Var  = a submatrix of the parameter VC matrix
; Fill (missing observations)
; Wts  = weighting variable
; Printvc

are available as usual. If your stratification indicators are set up properly for out of sample observations; ; Fill will allow you to extrapolate outside the estimation sample.

**WARNING:** If you do not have a stratification indicator already in use; ; Fill will not work. The _stratum_ variable is set up only for the estimation sample. Thus, with ; Pds = T, you cannot extrapolate outside the sample.

Two full sets of estimates are computed by this estimator:

1. **Constrained Least Squares Regression:** The fixed effects model above with all of the individual specific constants assumed equal is \( y_{it} = \alpha + \beta'x_{it} + \varepsilon_{it} \). This model is estimated by simple ordinary least squares. It is reported as ‘OLS Without Group Dummy Variables.’

2. **Least Squares Dummy Variable:** The fixed effects model with individual specific constant terms is estimated by partitioned ordinary least squares. For the one factor models, we formulate this model with \( N \) group specific constants and no overall constant.

**NOTE:** If you have time invariant regressors, such as sex or region, you cannot compute the fixed effects estimator. (The fixed effects estimator requires that there be within group variation in all variables for at least some groups.) In this case, you should use ; Random in your command.

**Program Output for One Way Fixed Effects Models**

For purposes of the discussion, define the four models:

- **Model 1** \( y_{it} = \alpha + \varepsilon_{it} \) (no group effects or xs),
- **Model 2** \( y_{it} = \alpha_i + \varepsilon_{it} \) (group dummies only),
- **Model 3** \( y_{it} = \alpha + \beta'x_{it} + \varepsilon_{it} \) (regressors only),
- **Model 4** \( y_{it} = \alpha_i + \beta'x_{it} + \varepsilon_{it} \) (xs and group effects).

Output from this program, in the order in which it will appear, is as follows:

1. Ordinary least squares regression of \( y \) on a single constant and the regressors, \( x_1,...,x_K \). *These K variables do not include one.* This is Model 3 above. Output consists of the standard results for least squares regression. The diagnostic statistics in this regression output will also include the unconditional analysis of variance for the dependent variable. This is the usual ANOVA for the groups, ignoring the regressors. (See Section E11.5.1 for details.) The output from this procedure could be used to test the hypothesis that the unconditional mean of \( y \) is the same in all groups. (This test is done by the program. See
part 3 below.)

2. Ordinary least squares estimates of Model 4 above. Output is the same as in part 1, the usual for a least squares regression. The estimates of the dummy variable coefficients and the estimated standard errors are listed in the output file if requested with ; Output = 2. (There may be hundreds or thousands of them!)

3. Test statistics for the various classical models. The table contains
   a. For Models 1 - 4, the log likelihood function, sum of squared residuals based on the least squares estimates, and $R^2$.
   b. Chi squared statistics based on the likelihood functions and F statistics based on the sums of squares for testing the restrictions of:
      - Model 1 as a restriction on Model 2 (no group effects on the mean of $y$),
      - Model 1 as a restriction on Model 3 (no fit in the regression of $y$ on $x$s),
      - Model 1 as a restriction on Model 4 (no group effects or fit in regression),
      - Model 2 as a restriction on Model 4 (group effects but no fit in regression),
      - Model 3 as a restriction on Model 4 (fit in regression but no group effects).

The statistic, degrees of freedom, and prob value (probability that the statistic would be equalled or exceeded by the chi squared or F random variable) are given for each hypothesis.

**Saved Results**

As always, the matrices $b$ and $varb$ are saved by the procedure. In addition, a matrix $alphafe$ will contain the estimates of the fixed effects, $\alpha_i$. This matrix is limited to 20,000 cells, so if your data have more than 20,000 groups, $alphafe$ will contain the first 20,000 fixed effects computed. Estimates of the variances or standard errors of the fixed effects are not kept. But, a simple method of computing them is given below.

Scalars that are kept are:

- $ssqrd$ = $s^2$ from least squares dummy variable (LSDV)
- $rsqrd$ = $R^2$ from LSDV
- $s$ = $\sqrt{s^2}$ from LSDV
- $sumsqdev$ = sum of squared residuals from LSDV
- $rho$ = estimated disturbance autocorrelation from whatever model is fit last
- $degfrdm$ = $\sum T_i - K$
- $sy$ = standard deviation of Lhs variable
- $ybar$ = mean of Lhs variable
- $kreg$ = $K$
- $nreg$ = total number observations
- $logl$ = log likelihood from LSDV model
- $exitcode$ = 0.0 if the model was estimable
- $ngroup$ = number of groups
- $nperiod$ = number of periods. This will be 0.0 if you fit a one way model.
The Last Model is constructed as usual, \textit{b\_variable}. Predicted values are based on the last model estimated, one or two way, fixed or random. Predictions are not listed when you use the group means estimator, but they can be computed with \textbf{MATRIX}.

**Robust Estimation of the Fixed Effects OLS Covariance Matrix**

There is a counterpart to the White estimator for unspecified heteroscedasticity for the one way fixed effects model. The model is

\[ y_{it} = \alpha_i + \beta' x_{it} + \epsilon_{it}. \]

Suppose that every \( \epsilon_{it} \) has a different variance, \( \sigma^2_{it} \). In the fashion of White’s estimator for the linear model, the natural approach is simply to replace \( \epsilon_{it}^2 \) with \( e_{it}^2 \) in the preceding, and compute

\[
\text{Est. Asy. Var}[b_{\text{lsdv}}] = [X^*'X^*]^{-1} X^* \hat{\Omega} X^* [X^*'X^*]^{-1}
\]

where ‘*’ denotes deviations from group means. This produces the same results as if the White correction were applied to the OLS results in full model including both regressors and group dummy variables.

If the variances can be assumed to be the same for all observations in the \( i \)th group, then each group specific variance can be estimated by the group mean squared residual, and the result inserted directly into the formulas above. In this case, \( \Omega \) becomes a block diagonal matrix, in which the \( i \)th diagonal block is \( \sigma^2_{i}I \). (This resembles the time series/cross section model discussed at the end of this chapter.) In practical terms, we simply replace \( \hat{e}_{it}^2 \) with \( s_i^2 \) in the estimate of the asymptotic covariance matrix. To request these estimators, add

\[
; \text{Het} \quad \text{or} \quad ; \text{Het} ; \text{Hc1} \quad \text{or} \quad ; \text{Het} ; \text{Hc2} \quad \text{or} \quad ; \text{Het} ; \text{Hc3}
\]

and

\[
; \text{Het} = \text{group}
\]

respectively to the \texttt{REGRESS} command.

The asymptotic covariance matrix for the fixed effects estimator may also be estimated with a Newey-West style correction for autocorrelation. Request this computation with

\[
; \text{Lags} = \text{the number of lags, up to 10}
\]

### 14.6.4 One Way Random Effects Model

The command for the random effects model is

\texttt{REGRESS ; Lhs = dependent variable}
\texttt{; Rhs = independent variables}
\texttt{; Str = stratification or Pds = count (variable)}
\texttt{; Panel}
\texttt{; Random Effects $}

This is the base case. Other specifications will be added as we proceed.
NOTE: The default command for panel data regressions is simply

\[
\text{REGRESS ; Lhs = ... ; Rhs = ... ; Panel ; Str =... or ; Pds = ... }
\]

This command requests both FE and RE results. The full set of results for both models will be presented. Adding either ; Fixed Effects or ; Random Effects will suppress the display of results for the other model.

NOTE: In computing the random effects model, the second step FGLS estimator generally relies on the first step OLS and LSDV (fixed effects) sums of squares. You may be suppressing the FE model, perhaps because of the presence of time invariant variables which preclude the FE model, but not the RE model. In previous versions of LIMDEP, and in some other programs, this will force the estimator to rely on another device to estimate the variance components, typically a group means estimator. In the current version of LIMDEP, the FE model is computed in the background, whether reported or not. The sums of squares needed are obtainable even in the presence of time invariant variables. Thus, you will get the same results for the RE model whether or not you have allowed LIMDEP to report the fixed effects results.

A crucial element of the computation of the random effects model is the estimation of the variance components. You may supply your own values for $\sigma^2_e$ and $\sigma^2_u$. The specification is

\[
; \text{Var} = s2e,s2u
\]

This overrides all other computations. The values are checked for validity. A nonpositive value forces estimation to halt at that point.

**Results Reported by the Random Effects Estimator**

After display of any previous results, including ordinary least squares and the fixed effects estimator, a display such as the following:

```
+----------------------------------------+
| Random Effects Model: v(i,t) = e(i,t) + u(i) |
| Estimates: Var[e] = .135360D-02 |
| Var[u] = .578177D-02 |
| Corr[v(i,t),v(i,s)] = .810298 |
| Lagrange Multiplier Test vs. Model (3) = 3721.10 |
| ( 1 df, prob value = .000000) |
| (High values of LM favor FEM/REM over CR model.) |
| Baltagi-Li form of LM Statistic = 3721.10 |
| Fixed vs. Random Effects (Hausman) = 37.62 |
| ( 6 df, prob value = .000001) |
| (High (low) values of H favor FEM (REM).) |
| Sum of Squares = .843423D+01 |
| R-squared = .990075D+00 |
+----------------------------------------+
```

will be presented, followed by the standard form table of coefficient estimates, standard errors, etc. The results in the table are as follows:
1. Estimates of $\sigma^2_e$ and $\sigma^2_u$ based on the least squares dummy variable model residuals. These are used to estimate the variance components. Since there are some potential problems that can arise, the sequence of steps taken in this part is documented in the trace file. The application shows an example. This trace output may be quite lengthy, as several attempts may be made to fit the model with different variance components estimators.

2. The estimate of $\rho = \sigma^2_u / (\sigma^2_e + \sigma^2_u)$ based on whatever first round estimator has been used.

3. Breusch and Pagan’s Lagrange multiplier statistic for testing the REM against the simple linear regression model with no common effects is

$$LM = \frac{(\sum_{i=1}^N T_i)^2}{[2\sum_{i=1}^N T_i (T_i - 1)]} \left[ \frac{\sum_{i=1}^N (T_i \bar{e})^2}{\sum_{i=1}^N \hat{e}^2_{i}} - 1 \right]^2$$

$$= \frac{(\sum_{i=1}^N T_i)^2}{[2\sum_{i=1}^N T_i (T_i - 1)]} \left[ \frac{\sum_{i=1}^N [(T_i \bar{e})^2 - \hat{e}_{i}^2]}{\sum_{i=1}^N \hat{e}_{i}^2} \right]^2$$

4. Baltagi and Li’s modification of the LM statistic for unbalanced panels is also reported with the results. This replaces the term outside the square brackets with

$$B = \frac{\frac{1}{2} N^2 \left( \sum_{i=1}^N N / T_i \right)^2}{\sum_{i=1}^N \left( 1 / T_i^2 \right) - N \sum_{i=1}^N \left( 1 / T_i \right)}$$

$LIMDEP$’s computation of the LM statistic already accounts for unbalanced panels. The Baltagi and Li modification evidently has some improved finite sample performance. As can be seen in the example above, which uses a balanced panel, the Baltagi and Li correction has no impact when group sizes are all equal. In this case, both these complicated statistics reduce to $NT/[2(T-1)]$. The prob value is given for the LM statistic. To a degree of approximation, the same result would apply to the Baltagi and Li form.

5. Hausman’s chi squared statistic for testing the REM against the FEM is reported next.

$$H = (\hat{\beta}_{FE} - \hat{\beta}_{RE})' [\text{Est.Var}(\hat{\beta}_{FE}) - \text{Est.Var}(\hat{\beta}_{RE})]^{-1} (\hat{\beta}_{FE} - \hat{\beta}_{RE})$$

The prob value and degrees of freedom for the Hausman statistic are reported.

**HINT:** Large values of the Hausman statistic argue in favor of the fixed effects model over the random effects model. Large values of the LM statistic argue in favor of one of the one factor models against the classical regression with no group specific effects. A large value of the LM statistic in the presence of a large Hausman statistic (as in our application) argues in favor of the fixed effects model.
**NOTE:** Sometimes it is not possible to compute the Hausman statistic. The difference matrix in the formula above may not be positive definite. The theory does not guarantee this. It is more likely to be so, but still not certain, if the same estimate of $\sigma_e^2$ is used for both cases. As such, *LIMDEP* uses the FGLS estimator of this, however it has been obtained, for the computation. Still, the matrix may fail to be positive definite. In this case, a 0.00 is reported for the statistic and a diagnostic warning appears in the results. Users are warned, some other programs attempt to bypass this issue by using some other matrix or some other device to force a positive statistic. These ad hoc measures do not solve the problem – they merely mask it. At worst, the appropriate zero value can be replaced by a value that appears to be ‘significant.’

6. The simple sum of squared residuals based on the random effects coefficients is reported.

7. An $R^2$ measure is reported (by popular request)

$$R^2 = 1 - \frac{\sum_{i=1}^{N} \sum_{t=1}^{T} (y_{it} - \hat{\beta}_i x_{it})^2}{\sum_{i=1}^{N} \sum_{t=1}^{T} (y_{it} - \bar{y})^2}.$$  

Users are warned, this measure can be negative. It is only guaranteed to be positive when OLS has been used to fit a model with a constant term. There are other measures that could be computed, such as the squared correlation between the actual and fitted values, but neither these, nor the one above, are fit measures in the same sense as in the linear model.

8. Estimates of the coefficients, their standard errors, and the ratio of each coefficient to its estimated standard error. (This is asymptotically distributed as standard normal.)

As always, the matrices $b$ and $varb$ are saved by the procedure. These will be the FGLS estimates of the random effects model

- $ssqrd$ = $s^2$ from least squares dummy variable (LSDV) estimator or from FGLS
- $rsqrd$ = $R^2$ from LSDV
- $s$ = $\sqrt{s^2}$ from LSDV
- $sumsqdev$ = sum of squared residuals from LSDV
- $rho$ = estimated disturbance autocorrelation from whatever model is fit last,
- $degfrdm$ = $\Sigma T_i - K$
- $sy$ = standard deviation of Lhs variable
- $ybar$ = mean of Lhs variable
- $kreg$ = $K$
- $nreg$ = total number observations
- $logl$ = log likelihood from LSDV model
- $ssqrdu$ = estimate of $\sigma_u^2$ from FGLS
- $ssqrde$ = estimate of $\sigma_e^2$ from FGLS
- $ssqrdev$ = estimate of $\sigma_w^2$ from GLS if two way random effects model is fit
- $exitcode$ = 0.0 if the model was estimable
- $ngroup$ = number of groups
- $nperiod$ = number of periods. This will be 0.0 if you fit a one way model.

The *Last Model* is constructed as usual, *b_variable*. Predicted values are based on the last model estimated, one or two way, fixed or random.
Chapter 15

Models for Discrete Choice

15.1 Introduction

We define models in which the response variable being described is inherently discrete as qualitative response (QR) models. This chapter will describe two of LIMDEP’s many estimators for qualitative dependent variable model estimators. The simplest of these is the binomial choice models, which are the subject of Section 15.2. The ordered choice model in Section 15.3 is an extension of the binary choice model in which there are more than two ordered, nonquantitative outcomes, such as scores on a preference scale.

15.2 Modeling Binary Choice

A binomial response may be the outcome of a decision or the response to a question in a survey. Consider, for example, survey data which indicate political party choice, mode of transportation, occupation, or choice of location. We model these in terms of probability distributions defined over the set of outcomes. There are a number of interpretations of an underlying data generating process that produce the binary choice models we consider here. All of them are consistent with the models that LIMDEP estimates, but the exact interpretation is a function of the modeling framework.

The essential model command for the parametric binary choice models is

\[
\begin{align*}
\text{PROBIT} & \quad \text{or} \\
\text{LOGIT} & \\
\end{align*}
\]

; Lhs = dependent variable ; Rhs = regressors $

A latent regression is specified as

\[ y^* = \beta'x + \varepsilon. \]

The observed counterpart to \( y^* \) is

\[ y = 1 \text{ if and only if } y^* > 0. \]

This is the basis for most of the binary choice models in econometrics, and is described in further detail below. It is the same model as the reduced form in the previous paragraph. Threshold models, such as labor supply and reservation wages lend themselves to this approach.
The probabilities and density functions for the most common binary choice specifications are as follows:

**Probit**

\[
F = \int_{-\infty}^{x_i} \frac{\exp(-t^2/2)}{\sqrt{2\pi}} dt = \Phi(\beta'x_i), \quad f = \phi(\beta'x_i)
\]

**Logit**

\[
F = \frac{\exp(\beta'x_i)}{1 + \exp(\beta'x_i)} = \Lambda(\beta'x_i), \quad f = \Lambda(\beta'x_i)[1 - \Lambda(\beta'x_i)]
\]

### 15.2.1 Model Commands

The model commands for the five binary choice models listed above are largely the same:

```
PROBIT ; Lhs = dependent variable ; Rhs = regressors
or
LOGIT
```

*Data on the dependent variable may be either individual or proportions.* You need not make any special note of which. *LIMDEP* will inspect the data to determine which type of data you are using. In either case, you provide only a single dependent variable. As usual, you should include a constant term in the model unless your application specifically dictates otherwise.

### 15.2.2 Output

The binary choice models generate a very large amount of output. Computation begins with least squares estimation in order to obtain starting values.

**NOTE:** The OLS results will not normally be displayed in the output. To request the display, use ; OLS in any of the model commands.

### Reported Estimates

Final estimates include:

- \( \log L \) = the log likelihood function at the maximum,
- \( \log L_0 \) = the log likelihood function assuming all slopes are zero. If your Rhs variables do not include one, this statistic will be meaningless. It is computed as

\[
\log L_0 = n[P\log P + (1-P)\log(1-P)]
\]

where \( P \) is the sample proportion of ones.
• The chi squared statistic for testing $H_0: \beta = 0$ (not including the constant) and the significance level = probability that $\chi^2$ exceeds test value. The statistic is

$$\chi^2 = 2(\log L - \log L_0).$$

Numerous other results, listed in detail, will appear with these in the output. The standard statistical results, including coefficient estimates, standard errors, t ratios, and descriptive statistics for the Rhs variables appear next. A complete listing is given below with an example. After the coefficient estimates are given, two additional sets of results appear, an analysis of the model fit and an analysis of the model predictions.

We will illustrate with binary logit and probit estimates of a model for visits to the doctor using the German health care data described in Chapter E2. The first model command is

```
LOGIT ; Lhs = doctor ; Rhs = one,age,hhninc,hhkids,educ,married ; OLS
```

Note that the command requests the optional listing of the OLS starting values. The results for this command are as follows. With the exception of the table noted below, the same results (with different values, of course) will appear for all five parametric models. Some additional optional computations and results will be discussed later.

The initial OLS estimates are generally not reported unless requested with ; OLS.

```
+------------------------------------------------+
| Binomial logit model for binary choice         |
| These are the OLS values based on the          |
| binary variables for each outcome $Y(i) = j$.  |
+------------------------------------------------+
| Variable | Coefficient | Standard Error | b/St.Er. | P[|Z|>|z|] | Mean of X |
|----------|-------------|----------------|---------|---------|-----------|
| Constant | 0.56661068  | 0.02118790     | 26.742  | 0.000   | 43.5256898|
| AGE      | 0.0468710   | 0.0029114      | 16.099  | 0.000   | 43.5256898|
| HHNINC   | -0.03976003 | 0.01726565     | 2.303   | 0.021   | 11.3206310|
| HHKIDS   | -0.05217181 | 0.00680260     | -7.669  | 0.000   | 11.3206310|
| EDUC     | -0.01071245 | 0.00757540     | -1.569  | 0.016   | 11.3206310|
| MARRIED  | 0.01946888  | 0.00757540     | 2.570   | 0.010   | 11.3206310|

Standard results for maximum likelihood estimation appear next (or first if OLS is not presented). These are the results generated for all models fit by maximum likelihood. The Hosmer-Lemeshow chi squared statistic is specific to the binary choice models. It is discussed in Section E18.3.7. The information criteria are computed from the log likelihood, $\log L$, and the number of parameters estimated, $K$, as follows:

- **AIC** = Akaike Information Criterion

  $\text{AIC} = -2(\log L - K)/n$

- **BIC** = Bayesian Information Criterion

  $\text{BIC} = -2(\log L - K\log K)/n$

- **Finite Sample AIC**

  $\text{Finite Sample AIC} = -2(\log L - K - K(K+1)/(n-K-1))/n$

- **HQIC**

  $\text{HQIC} = -2(\log L - K\log(\log n))/n$
Normal exit from iterations. Exit status=0.

| Binomial Logit Model for Binary Choice |
| Maximum Likelihood Estimates           |
| Dependent variable              DOCTOR |
| Weighting variable               None    |
| Number of observations           27326  |
| Iterations completed               4     |
| Log likelihood function       -17673.10 |
| Number of parameters               6     |
| Info. Criterion: AIC =          1.29394 |
| Finite Sample: AIC =          1.29394  |
| Info. Criterion: BIC =          1.29574  |
| Info. Criterion: HQIC =          1.29452 |
| Restricted log likelihood      -18019.55 |
| McFadden Pseudo R-squared       .0192266 |
| Chi squared                    692.9077  |
| Degrees of freedom              5      |
| Prob[ChiSqd > value] =          .0000000 |
| Hosmer-Lemeshow chi-squared = 110.37153 |
| P-value= .00000 with deg.fr. =    8     |

| Variable | Coefficient | Standard Error | b/St.Er. | P[|Z|>z] | Mean of X |
|----------|-------------|----------------|----------|--------|----------|
| Constant | .25111543 | .09113537 | 2.755 | .0059 |
| AGE      | .02070863 | .00128517 | 16.114 | .0000 | 43.5256898 |
| HHNINC   | -.18592232 | .07506403 | -2.477 | .0133 | .35208362 |
| HHKIDS   | -.22947000 | .02953694 | -7.769 | .0000 | .40273000 |
| EDUC     | -.04558783 | .00564646 | -8.074 | .0000 | .11.3206310 |
| MARRIED  | .08529305 | .0328573 | 2.562 | .0104 | .75861817 |

The next set of results computes various fit measures for the model. This table of information statistics is produced only for the logit model. It is generally used for analysis of the generalized maximum entropy (GME) estimator of the multinomial logit model, but it also provides some useful information for the binomial model even when fit by ML instead of GME.

The entropy statistics are computed as follows:

\[
Entropy = - \sum_i P_i \log P_i
\]

where \( P_i \) is the probability predicted by the model. The three ‘models’ are ‘\( M \)’, the model fit by maximum likelihood, ‘\( MC \)’, the model in which all predicted probabilities are the sample proportion of ones (here 0.6291), and ‘\( M_0 \)’ (no model) in which all predicted probabilities are 0.5. The normalized entropy is the entropy divided by \( n \log 2 \). Finally, the entropy ratio statistic equals \( 2(n \log 2)(1 - \text{normalized entropy}) \). The percent correct predicted values are discussed below.

The next set of results examines the success of the prediction rule

\[
\text{Predict } y_i = 1 \text{ if } P_i > P^* \text{ and 0 otherwise}
\]

where \( P^* \) is a defined threshold probability. The default value of \( P^* \) is 0.5, which makes the prediction rule equivalent to ‘Predict \( y_i = 1 \) if the model says the predicted event \( y_i = 1 \mid x_i \) is more likely than the complement, \( y_i = 0 \mid x_i \)’. You can change the threshold from 0.5 to some other value with

\[
; \text{Limit} = \text{your } P^*
\]
A variety of fit measures for the model are listed.

```plaintext
| Fit Measures for Binomial Choice Model |
| Logit model for variable DOCTOR |
| Proportions P0= .370892  P1= .629108 |
| N = 27326  N0= 10135  N1= 17191 |
| LogL= -17673.098  LogL0= -18019.552 |
| Estrella = 1-(L/L0)^(-2L0/n) = .02528 |

<table>
<thead>
<tr>
<th>Efрон</th>
<th>McFadden</th>
<th>Ben./Lerman</th>
</tr>
</thead>
<tbody>
<tr>
<td>.02435</td>
<td>.01923</td>
<td>.54487</td>
</tr>
<tr>
<td>Cramer</td>
<td>Veall/Zim.</td>
<td>Rsqrd ML</td>
</tr>
<tr>
<td>.02470</td>
<td>.04348</td>
<td>.02504</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Information  Akaike I.C. Schwarz I.C.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Criteria 1.29394  1.29574</td>
</tr>
</tbody>
</table>

<p>| Predictions for Binary Choice Model. Predicted value is |
| 1 when probability is greater than .500000, 0 otherwise. |
| Note, column or row total percentages may not sum to |
| 100% because of rounding. Percentages are of full sample. |
| Actual | Predicted Value |</p>
<table>
<thead>
<tr>
<th>Value</th>
<th>0</th>
<th>1</th>
<th>Total Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>378 ( 1.4%)</td>
<td>9757 ( 35.7%)</td>
<td>10135 ( 37.1%)</td>
</tr>
<tr>
<td>1</td>
<td>394 ( 1.4%)</td>
<td>16797 ( 61.5%)</td>
<td>17191 ( 62.9%)</td>
</tr>
<tr>
<td>Total</td>
<td>772 ( 2.8%)</td>
<td>26554 ( 97.2%)</td>
<td>27326 (100.0%)</td>
</tr>
</tbody>
</table>
```
This table computes a variety of conditional and marginal proportions based on the results using the defined prediction rule. For examples, the 97.708\% equals (16797/17191)100\% while the 63.256\% is (16797/26554)100\%.

---

Analysis of Binary Choice Model Predictions Based on Threshold = .5000

Prediction Success

<table>
<thead>
<tr>
<th>Prediction Success</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Sensitivity = actual 1s correctly predicted</td>
<td>97.708%</td>
</tr>
<tr>
<td>Specificity = actual 0s correctly predicted</td>
<td>3.730%</td>
</tr>
<tr>
<td>Positive predictive value = predicted 1s that were actual 1s</td>
<td>63.256%</td>
</tr>
<tr>
<td>Negative predictive value = predicted 0s that were actual 0s</td>
<td>48.964%</td>
</tr>
<tr>
<td>Correct prediction = actual 1s and 0s correctly predicted</td>
<td>62.852%</td>
</tr>
</tbody>
</table>

Prediction Failure

<table>
<thead>
<tr>
<th>Prediction Failure</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>False pos. for true neg. = actual 0s predicted as 1s</td>
<td>96.270%</td>
</tr>
<tr>
<td>False neg. for true pos. = actual 1s predicted as 0s</td>
<td>2.292%</td>
</tr>
<tr>
<td>False pos. for predicted pos. = predicted 1s actual 0s</td>
<td>36.744%</td>
</tr>
<tr>
<td>False neg. for predicted neg. = predicted 0s actual 1s</td>
<td>51.036%</td>
</tr>
<tr>
<td>False predictions = actual 1s and 0s incorrectly predicted</td>
<td>37.148%</td>
</tr>
</tbody>
</table>

---

Retained Results

The results saved by the binary choice models are:

Matrices: 
- \( b \) = estimate of \( \beta \) (also contains \( \gamma \) for the Burr model)
- \( varb \) = asymptotic covariance matrix

Scalars: 
- \( kreg \) = number of variables in Rhs
- \( nreg \) = number of observations
- \( logl \) = log likelihood function

15.2.3 Analysis of Marginal Effects

Marginal effects in a binary choice model may be obtained as

\[
\frac{\partial E[y \mid x]}{\partial x} = \frac{\partial F(\beta'x)}{\partial \beta} = \frac{dF(\beta'x)}{d(\beta'x)} \beta = f(\beta'x)\beta
\]

That is, the vector of marginal effects is a scalar multiple of the coefficient vector. The scale factor, \( f(\beta'x) \), is the density function, which is a function of \( x \). (The densities for the five binary choice models are listed in Section E18.3.1.) This function can be computed at any data vector desired. You can request the computation to be done automatically at the vector of means of the current sample by adding

\verbatim
; Marginal Effects
\endverbatim

to your command.
Marginal Effects for Dummy Variables

When one of the variables in \( x \) is a dummy variable, the derivative approach to estimating the marginal effect is not appropriate. An alternative which is closer to the desired computation for a dummy variable which we denote \( z \), is

\[
\Delta F_z = \text{Prob}[y = 1 \mid z = 1] - \text{Prob}[y = 1 \mid z = 0] = F(\beta'x + \alpha z \mid z = 1) - F(\beta'x + \alpha z \mid z = 0).
\]

For this type of variable, the asymptotic standard error must be changed as well. This is accomplished simply by changing the appropriate row of \( G \) to

\[
G_z = \begin{bmatrix} f(\beta'x + \alpha z) \\ 1 \end{bmatrix}' - \begin{bmatrix} f(\beta'x + \alpha z) \\ 0 \end{bmatrix}'.
\]

LIMDEP examines the variables in the model and makes this adjustment automatically.

15.2.4 Robust Covariance Matrix Estimation

The preceding describes a covariance estimator that accounts for a specific, observed aspect of the data. The concept of the ‘robust’ covariance matrix is that it is meant to account for hypothetical, unobserved failures of the model assumptions. The intent is to produce an asymptotic covariance matrix that is appropriate even if some of the assumptions of the model are not met. (It is an important, but infrequently discussed issue whether the estimator, itself, remains consistent in the presence of these model failures – that is, whether the so called robust covariance matrix estimator is being computed for an inconsistent estimator.) (Section R10.8 in the Reference Guide provides general discussion of robust covariance matrix estimation.)

The Sandwich Estimator

It is becoming common in the literature to adjust the estimated asymptotic covariance matrix for possible misspecification in the model which leaves the MLE consistent but the estimated asymptotic covariance matrix incorrectly computed. One example would be a binary choice model with unspecified latent heterogeneity. A frequent adjustment for this case is the ‘sandwich estimator,’ which is the choice based sampling estimator suggested above with weights equal to one. (This suggests how it could be computed.) The desired matrix is

\[
\text{Est. Asy. Var} \left[ \hat{\beta} \right] = \sum_{i=1}^{n} \begin{bmatrix} \frac{\partial^2 \log F_i}{\partial \hat{\beta} \partial \hat{\beta}'} \\ \frac{\partial \log F_i}{\partial \hat{\beta}} \left( \frac{\partial \log F_i}{\partial \hat{\beta}'} \right) \end{bmatrix}^{-1} \begin{bmatrix} \sum_{i=1}^{n} \left( \frac{\partial \log F_i}{\partial \hat{\beta}} \right) \left( \frac{\partial \log F_i}{\partial \hat{\beta}'} \right) \\ \sum_{i=1}^{n} \left( \frac{\partial^2 \log F_i}{\partial \hat{\beta} \partial \hat{\beta}'} \right) \end{bmatrix}^{-1}
\]

Three ways to obtain this matrix are

; Wts = one ; Choice Based sampling
or
; Robust
or
; Cluster = 1
The computation is identical in all cases. (As noted below, the last of them will be slightly larger, as it will be multiplied by $n/(n-1)$.)

**Clustering**

A related calculation is used when observations occur in groups which may be correlated. This is rather like a panel; one might use this approach in a random effects kind of setting in which observations have a common latent heterogeneity. The parameter estimator is unchanged in this case, but an adjustment is made to the estimated asymptotic covariance matrix. The calculation is done as follows: Suppose the $n$ observations are assembled in $G$ clusters of observations, in which the number of observations in the $i$th cluster is $n_i$. Thus, $\sum_{i=1}^{G} n_i = n$. Let the observation specific gradients and Hessians be

$$ g_{ij} = \frac{\partial \log L_{ij}}{\partial \beta} \quad H_{ij} = \frac{\partial^2 \log L_{ij}}{\partial \beta \partial \beta'}.$$

The uncorrected estimator of the asymptotic covariance matrix based on the Hessian is

$$V_H = -H^{-1} = \left( -\sum_{i=1}^{G} \sum_{j=1}^{n_i} H_{ij} \right)^{-1}$$

Estimators for some models will use the BHHH estimator, instead;

$$V_B = \left[ \sum_{i=1}^{G} \sum_{j=1}^{n_i} g_{ij} g_{ij}' \right]^{-1}$$

Let $V$ be the estimator chosen. Then, the corrected asymptotic covariance matrix is

$$\text{Est. Asy. Var}\left[ \hat{\beta} \right] = V \frac{G}{G-1} \left[ \sum_{i=1}^{G} \left( \sum_{j=1}^{n_i} g_{ij} \right) \left( \sum_{j=1}^{n_i} g_{ij}' \right) \right] V$$

Note that if there is exactly one observation per cluster, then this is $G/(G-1)$ times the sandwich estimator discussed above. Also, if you have fewer clusters than parameters, then this matrix is singular – it has rank equal to the minimum of $G$ and $K$, the number of parameters.

To request the estimator, your command must include

; Cluster = specification

where the specification is either the fixed value if all the clusters are the same size, or the name of an identifying variable if the clusters vary in size. Note, this is not the same as the variable in the Pds function that is used to specify a panel. The cluster specification must be an identifying code that is specific to the cluster. For example, our health care data used in our examples is an unbalanced panel. The first variable is a family id, which we will use as follows

; Cluster = id
15.3 Ordered Choice Models

The basic ordered choice model is based on the following specification: There is a latent regression,
\[ y_i^* = \beta'x_i + \varepsilon_i, \quad \varepsilon_i \sim F(\varepsilon_i | \theta), \quad E[\varepsilon_i|x] = 0, \quad \text{Var}[\varepsilon_i|x] = 1, \]

The observation mechanism results from a complete censoring of the latent dependent variable as follows:
\[ y_i = \begin{cases} 0 & \text{if } y_i \leq \mu_0, \\ 1 & \text{if } \mu_0 < y_i \leq \mu_1, \\ 2 & \text{if } \mu_1 < y_i \leq \mu_2, \\ \vdots & \text{if } \mu_{J-1} < y_i \leq \mu_J, \\ J & \text{if } y_i > \mu_{J-1}. \end{cases} \]

The latent ‘preference’ variable, \( y_i^* \) is not observed. The observed counterpart to \( y_i^* \) is \( y_i \). Four stochastic specifications are provided for the basic model shown above. The ordered probit model applies in applications such as surveys, in which the respondent expresses a preference with the above sort of ordinal ranking. The variance of \( \varepsilon_i \) is assumed to be one, since as long as \( y_i^*, \beta \), and \( \varepsilon_i \) are unobserved, no scaling of the underlying model can be deduced from the observed data. Since the \( \mu_s \) are free parameters, there is no significance to the unit distance between the set of observed values of \( y \). They merely provide the coding. Estimates are obtained by maximum likelihood. The probabilities which enter the log likelihood function are
\[ \text{Prob}[y_i = j] = \text{Prob}[y_i^* \text{ is in the } j\text{th range}]. \]

The model may be estimated either with individual data, with \( y_i = 0, 1, 2, \ldots \) or with grouped data, in which case each observation consists of a full set of \( J+1 \) proportions, \( p_0, \ldots, p_J \).

**NOTE:** If your data are not coded correctly, this estimator will abort with one of several possible diagnostics – see below for discussion. Your dependent variable must be coded 0, 1, 2, ..., \( J \). We note that this differs from some other econometric packages which use a different coding convention.

There are numerous variants and extensions of this model which can be estimated: The underlying mathematical forms are shown below, where the CDF is denoted \( F(z) \) and the density is \( f(z) \). (Familiar synonyms are given as well.)

**Probit:**
\[ F(z) = \int_{-\infty}^{z} \frac{\exp(-t^2/2)}{\sqrt{2\pi}} dt = \Phi(z), \quad f(z) = \phi(z) \]

**Logit:**
\[ F(z) = \frac{\exp(z)}{1 + \exp(z)} = \Lambda(z), \quad f(z) = \Lambda(z)[1 - \Lambda(z)] \]

The ordered probit model is an extension of the probit model for a binary outcome with normally distributed disturbances. The ordered logit model results from the assumption that \( \varepsilon \) has a standard logistic distribution instead of a standard normal.
15.3.1 Estimating Ordered Probability Models

The essential command for estimating ordered probability models is

```
ORDERED ; Lhs = y ; Rhs = regressors $
```

If you are using individual data, the Lhs variable must be coded 0,1,...,J. All the values must be present in the data. LIMDEP will look for empty cells. If there are any, estimation is halted. (If value ‘j’ is not represented in the data, then the threshold parameter, \( \mu_j \) is not estimable.) In this circumstance, you will receive a diagnostic such as

```
ORDE,Panel,BIVA PROBIT:A cell has (almost) no observations.
Empty cell: Y never takes value 2
```

This diagnostic means exactly what it says. The ordered probability model cannot be estimated unless all cells are represented in the data. Users frequently overlook the coding requirement, \( y = 0,1,... \). If you have a dependent variable that is coded 1,2,..., you will see the following diagnostic:

```
Models - Insufficient variation in dependent variable.
```

The reason this particular diagnostic shows up is that LIMDEP creates a new variable from your dependent variable, say \( y \), which equals zero when \( y \) equals zero, and one when \( y \) is greater than zero. It then tries to obtain starting values for the model by fitting a regression model to this new variable. If you have miscoded the Lhs variable, the transformed variable always equals one, which explains the diagnostic. In fact, there is no variation in the transformed dependent variable. If this is the case, you can simply use `CREATE` to subtract 1.0 from your dependent variable to use this estimator.

The probit model is the default specification. To estimate an ordered logit, add

```
; Model = Logit
```

to the command. The standardized logistic distribution (mean zero, standard deviation approximately 1.81) is used as the basis of the model instead of the standard normal.

15.3.2 Model Structure and Data

This model must include a constant term, `one`, as the first Rhs variable. Since the equation does include a constant term, one of the \( \mu \)s is not identified. We normalize \( \mu_0 \) to zero. (Consider the special case of the binary probit model with something other than zero as its threshold value. If it contains a constant, this cannot be estimated.) Data may be grouped or individual. (Survey data might logically come in grouped form.) If you provide individual data, the dependent variable is coded 0, 1, 2, ..., J. There must be at least three values. Otherwise, the binary probit model applies. If the data are grouped, a full set of proportions, \( p_0, p_1, ..., p_J \), which sum to one at every observation must be provided. In the individual data case, the data are examined to determine the value of J, which will be the largest observed value of \( y \) which appears in the sample. In the grouped data case, \( J \) is one less than the number of Lhs variables you provide. Once again, we note that other programs sometimes use different normalizations of the model. For example, if the constant term is
forced to equal zero, then one will instead, add a nonzero threshold parameter, μ0, which equals zero in the presence of a nonzero constant term.

15.3.3 Output from the Ordered Probability Estimators

All of the ordered probit/logit models begin with an initial set of least squares results of some sort. These are suppressed unless your command contains ; OLS. The iterations are then followed by the maximum likelihood estimates in the usual tabular format. The final output includes a listing of the cell frequencies for the outcomes. When the data are stratified, this output will also include a table of the frequencies in the strata. The log likelihood function, and a log likelihood computed assuming all slopes are zero are computed. For the latter, the threshold parameters are still allowed to vary freely, so the model is simply one which assigns each cell a predicted probability equal to the sample proportion. This appropriately measures the contribution of the nonconstant regressors to the log likelihood function. As such, the chi squared statistic given is a valid test statistic for the hypothesis that all slopes on the nonconstant regressors are zero.

The sample below shows the standard output for a model with six outcomes. These are the German health care data described in detail in Chapter E2. The dependent variable is the self reported health satisfaction rating. For the purpose of a convenient sample application, we have truncated the health satisfaction variable at five by discarding observations – in the original data set, it is coded 0,1,...,10.

<table>
<thead>
<tr>
<th>Ordered Probability Model</th>
<th>Maximum Likelihood Estimates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dependent variable</td>
<td>NEWHSAT</td>
</tr>
<tr>
<td>Weighting variable</td>
<td>None</td>
</tr>
<tr>
<td>Number of observations</td>
<td>8140</td>
</tr>
<tr>
<td>Log likelihood function</td>
<td>-11284.69</td>
</tr>
<tr>
<td>Number of parameters</td>
<td>9</td>
</tr>
<tr>
<td>Info. Criterion: AIC =</td>
<td>2.77486</td>
</tr>
<tr>
<td>Info. Criterion: BIC =</td>
<td>2.78261</td>
</tr>
<tr>
<td>Restricted log likelihood</td>
<td>-11308.02</td>
</tr>
<tr>
<td>McFadden Pseudo R-squared</td>
<td>0.0020635</td>
</tr>
<tr>
<td>Chi squared</td>
<td>46.66728</td>
</tr>
<tr>
<td>Degrees of freedom</td>
<td>4</td>
</tr>
<tr>
<td>Prob[ChiSqd &gt; value] =</td>
<td>0.0000000</td>
</tr>
<tr>
<td>Underlying probabilities</td>
<td>based on Normal</td>
</tr>
<tr>
<td>Cell frequencies for</td>
<td>outcomes</td>
</tr>
<tr>
<td>Y Count Freq Y Count Freq Y Count Freq</td>
<td></td>
</tr>
<tr>
<td>0 447 .054 1 255 .031 2 642 .078</td>
<td></td>
</tr>
<tr>
<td>3 1173 .144 4 1390 .170 5 4233 .520</td>
<td></td>
</tr>
</tbody>
</table>

| Variable | Coefficient | Standard Error | b/St.Er. | P[|Z|>z] | Mean of X |
|---------|-------------|----------------|----------|--------|----------|
| Constant| 1.32892012  | .07275667      | 18.265   | .0000  |
| FEMALE  | .04525825   | .02546350      | 1.777    | .0755  | .52936118 |
| HHNINC  | .35589979   | .07831928      | 4.544    | .0000  | .32998942 |
| HHKIDS  | .10603682   | .02664775      | 3.979    | .0001  | .33169533 |
| EDUC    | .00927669   | .00629721      | 1.473    | .1407  | 10.8759203 |

| Variable | Coefficient | Standard Error | b/St.Er. | P[|Z|>z] | Mean of X |
|---------|-------------|----------------|----------|--------|----------|
| Mu(1)   | .23634786   | .01236704      | 19.111   | .0000  |
| Mu(2)   | .62954428   | .01439990      | 43.719   | .0000  |
| Mu(3)   | 1.10763798  | .01405938      | 78.783   | .0000  |
| Mu(4)   | 1.55676227  | .01527126      | 101.941  | .0000  |
The model output is followed by a \((J+1) \times (J+1)\) frequency table of predicted versus actual values. (This table is not given when data are grouped or when there are more than 10 outcomes.) The predicted outcome for this tabulation is the one with the largest predicted probability.

| Cross tabulation of predictions. Row is actual, column is predicted. |
| Model = Probit. Prediction is number of the most probable cell. |

<table>
<thead>
<tr>
<th>Actual</th>
<th>Row Sum</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>447</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>447</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>255</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>255</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>642</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>642</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1173</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1173</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1390</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1390</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>4233</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4233</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Col Sum</td>
<td>8140</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8140</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Even though the model appears to be highly significant, the table of predictions has some large gaps in it. The estimation criterion for the ordered probability model is unrelated to its ability to predict those cells, and you will rarely see a predictions table that closely matches the actual outcomes. It often happens that even in a set of results with highly significant coefficients, only one or a few of the outcomes are predicted by the model.

Computation of predictions and ancillary variables is as follows: For each observation, the predicted probabilities for all \(J+1\) outcomes are computed. Then if you request \(; \text{List}\), the listing will contain

- \(\text{Predicted } Y\) is the \(Y\) with the largest probability.
- \(\text{Residual}\) is the largest of the \(J+1\) probabilities (i.e., \(\text{Prob}\[y = \text{fitted } Y]\)).
- \(\text{Var1}\) is the estimate of \(\mathbb{E}[Y] = \sum_{i=0}^{J} i \times \text{Prob}[Y_i = i]\).

(Nota note: since the outcomes are only ordinal, this is not a true expected value.)

- \(\text{Var2}\) is the probability estimated for the observed \(Y\).

Estimation results kept by the estimator are as follows:

- **Matrices:** \(b\) = estimate of \(\beta\), \(\text{varb}\) = estimated asymptotic covariance, \(\mu\) = \(J\)-1 estimated \(\mu\)s.

- **Scalars:** \(kreg\), \(nreg\), and \(logl\).

- **Last Model:** The labels are \(b\_variables\), \(mu1\), ...

The specification \(; \text{Par}\) adds \(\mu\) (the set of estimated threshold values) to \(b\) and \(\text{varb}\). The additional matrix, \(\mu\) is kept regardless, but the estimated asymptotic covariance matrix is lost unless the command contains \(; \text{Par}\).
Marginal effects in the ordered probability models are quite involved. Since there is no meaningful conditional mean function to manipulate, we consider, instead, the effects of changes in the covariates on the cell probabilities. These are:

\[
\frac{\partial \text{Prob}[\text{cell } j]}{\partial x_i} = [f(\mu_{j,1} - \beta'x_i) - f(\mu_{j} - \beta'x_i)] \times \beta,
\]

where \(f(.)\) is the appropriate density for the standard normal, \(\phi(\cdot)\), logistic density, \(\Lambda(\cdot)(1-\Lambda(\cdot))\), Weibull or Gompertz. Each vector is a multiple of the coefficient vector. But it is worth noting that the magnitudes are likely to be very different. In at least one case, Prob[cell 0], and probably more if there are more than three outcomes, the partial effects have exactly the opposite signs from the estimated coefficients. Thus, in this model, it is important to consider carefully the interpretation of the coefficient estimates. Marginal effects for all cells can be requested by including `Marginal Effects` in the command. An example appears below.

NOTE: This estimator segregates dummy variables for separate computation in the marginal effects. The marginal effect for a dummy variable is the simple difference of the two probabilities, with and without the variable. See the application below for an illustration.

```
+----------------------------------------------------+
| Marginal effects for ordered probability model     |
| M.E.s for dummy variables are Pr[y|x=1]-Pr[y|x=0]  |
| Names for dummy variables are marked by *          |
+----------------------------------------------------+
|Variable| Coefficient  | Standard Error |b/St.Er.|P[|Z|>z]| Mean of X|
|---------+--------------+----------------+--------+--------+----------|
|Constant|       .000000    ......(Fixed Parameter)........  |
|*FEMALE |    -.00498024       .00280960    -1.773   .0763  .52936118  |
|HHNINC  |    -.03907462       .00862973    -4.528   .0000  .32998942  |
|*HHKIDS |    -.01131976       .00277405    -4.081   .0000  .33169533  |
|EDUC    |    -.00101850       .00069179    -1.472   .1409  10.8759203  |
|---------+These are the effects on Prob[Y=00] at means.  |
|Constant|       .000000    ......(Fixed Parameter)........  |
|*FEMALE |    -.00209668       .00118069    -1.776   .0758  .52936118  |
|HHNINC  |    -.01647123       .00362630    -4.542   .0000  .32998942  |
|*HHKIDS |    -.00483428       .00119623    -4.041   .0001  .33169533  |
|EDUC    |    -.00042933       .00029148    -1.473   .1408  10.8759203  |
|---------+These are the effects on Prob[Y=01] at means.  |
|Constant|       .000000    ......(Fixed Parameter)........  |
|*FEMALE |     .01803285       .01014562     1.777   .0755  .52936118  |
|HHNINC  |     .14180876       .00073836    192.060   .0000  .32998942  |
|*HHKIDS |     .04218672       .00029837    141.390   .0000  .33169533  |
|EDUC    |     .00369631       .00250467     1.476   .1400  10.8759203  |
|--------+Effects for Y=02, Y=03 and Y=04 are omitted. |
|---------+These are the effects on Prob[Y=05] at means.  |
|Constant|       .000000    ......(Fixed Parameter)........  |
|*FEMALE |     .01803285       .01014562     1.777   .0755  .52936118  |
|HHNINC  |     .14180876       .00073836    192.060   .0000  .32998942  |
|*HHKIDS |     .04218672       .00029837    141.390   .0000  .33169533  |
|EDUC    |     .00369631       .00250467     1.476   .1400  10.8759203  |
+-------------------------------------------------------------------------+
| Summary of Marginal Effects for Ordered Probability Model (probit)      |
|-------------------------------------------------------------------------+
|Variable| Y=00  Y=01  Y=02  Y=03  Y=04  Y=05  Y=06  Y=07 |
|---------+--------+--------+--------+--------+--------+--------+--------+--------|
|ONE      | .0000  .0000  .0000  .0000  .0000  .0000  .0000  .0000  |
|*FEMALE  | -.0050  -.0021  -.0041  -.0047  -.0021  .0180  |
|HHNINC   | -.0391  -.0165  -.0326  -.0373  -.0164  .1418  |
|*HHKIDS  |  -.0113  -.0048  -.0096  -.0112  -.0052  .0422  |
|EDUC     |  -.0010  -.0004  -.0008  -.0010  -.0004  .0037  |
# Chapter 16

## Censoring and Sample Selection

### 16.1 Introduction

The models described in this chapter are variations on the following general structure:

**Latent Underlying Regression:**
\[ y_i^* = \beta'x_i + \varepsilon_i, \varepsilon_i \sim N[0,\sigma^2]. \]

**Observed Dependent Variable:**
- If \( y_i^* \leq L_i \), then \( y_i = L_i \) (lower tail censoring)
- If \( y_i^* \geq U_i \), then \( y_i = U_i \) (upper tail censoring)
- If \( L_i < y_i^* < U_i \), then \( y_i = y_i^* = \beta'x_i + \varepsilon_i \).

The thresholds, \( L_i \) and \( U_i \), may be constants or variables. We accommodate censoring in the upper or lower (or both) tails of the distribution. The most familiar case of this model in the literature is the ‘tobit’ model, in which \( U_i = +\infty \) and \( L_i = 0 \), i.e., the case in which the observed data contain a cluster of zeros. In the standard ‘censored regression,’ or tobit model, the censored range of \( y_i^* \) is the half of the line below zero. (For convenience, we will drop the observation subscript at this point.) If \( y^* \) is not positive, a zero is observed for \( y \), otherwise the observation is \( y^* \). Models of expenditure are typical. We also allow censoring of the upper tail (‘on the right’). A model of the demand for tickets to sporting events might be an application, since the actual demand is only observed if it is not more than the capacity of the facility (stadium, etc.). A somewhat more elaborate specification is obtained when the range of \( y^* \) is censored in both tails. This is the ‘two limit probit’ model. An application might be a model of weekly hours worked, in which less than half time is reported as 20 and more than 40 is reported as ‘full time,’ i.e., 40 or more.

**NOTE:** The mere presence of a clump of zeros in the data set does not, by itself, adequately motivate the tobit model. The specification of the model also implies that the nonlimit observations will have a continuous distribution with observations near the limit points. In general, if you try to fit a tobit model, e.g., to financial data in which there is a clump of zeros, and the nonzero observations are ordinary financial variables far from zero, the model is as likely as not to break down during estimation. In such a case, the model of sample selection is probably a more appropriate specification.

### 16.2 Single Equation Tobit Regression Model

The base case considered here is the familiar ‘tobit’ model:

**Latent underlying regression:**
\[ y_i^* = \beta'x_i + \varepsilon_i, \varepsilon_i \sim N[0,\sigma^2]. \]

**Observed dependent variable:**
- If \( y_i^* \leq L_i \), then \( y_i = L_i \) (lower tail censoring)
- If \( y_i^* \geq U_i \), then \( y_i = U_i \) (upper tail censoring)
- If \( L_i < y_i^* < U_i \), then \( y_i = y_i^* = \beta'x_i + \varepsilon_i \).

Within this framework, the most familiar form is the lower censoring only, at zero variant.
16.2.1 Commands

The basic command for estimation of the censored regression, or tobit model is

```
TOBIT ; Lhs = y ; Rhs = ... 
```

The default value for the censoring limit is zero, at the left (i.e., the familiar case). Censoring limits can be varied in two fashions. To specify upper, rather than lower tail censoring, add

```
; Upper
```

to the model. With no other changes, this would specify a model in which the observed values of the dependent variable would be either zero or negative rather than zero or positive. The specific limit point to use can be changed by using

```
; Limit = limit value
```

where ‘limit value’ is either a fixed value (number or scalar) or the name of a variable. For example, the model of the demand for sporting events at stadiums with fixed capacities which sell out a significant proportion of the time might be

```
TOBIT ; Lhs = tickets ; Rhs = one, price, ... ; Upper censoring ; Limit = capacity 
```

Models with censoring in both tails of the distribution are requested by changing the ; Limit specification to

```
; Limits = lower limit, upper limit
```

where ‘lower limit’ and ‘upper limit’ are either numbers, scalars, or the names of variables (or one of each). For example, in a labor supply model, we might have

```
; Limits = 20,40
```

Other options for the tobit model are the standard ones for nonlinear models, including

```
; Printvc to display the estimated asymptotic covariance matrix
; List to display predicted values
; Parameters to include the estimate of $\sigma$ in the retained parameter vector
; Maxit = n to set maximum iterations
; Alg = name to select algorithm
; Tlf, ; Tlb, ; Tlg to set the convergence criteria
    (use ; Set to keep these settings)
; Output = value to control the technical output during iterations
; Keep = name to retain fitted values
; Res = name to retain residuals
; Marginal Effects
```
and so on. Sample clustering for the estimated asymptotic covariance matrix may be requested with

; Cluster = specification

as usual.

16.2.2 Results for the Tobit Model

You may request the display of ordinary least squares results by adding

; OLS

to the command. These will be suppressed if you do not include this request. The OLS values will be used as the starting values for the iterations. Maximum likelihood estimates are presented in full. Note that unlike most of the discrete choice models, there is no restricted log likelihood presented. The maximum likelihood estimates for a model that contains only a constant term are no less complicated than one with covariates, and there is no closed form solution for the \((\beta, \sigma)\) parameter pair for this model. For a general test of the joint significance of all the variables in the model, we suggest the standard trio of Neyman-Pearson tests, which can be carried out as follows: First set up the Rs2 variables in the model.

\begin{verbatim}
NAMELIST ; vars = the x variables in the model, without the constant term $ 
CALC ; kx = Col(vars) $ 
TOBIT ; Lhs = y ; Rhs = one $ 
CALC ; l0 = logl $ 
\end{verbatim}

This command will produce the Lagrange multiplier statistic.

\begin{verbatim}
TOBIT ; Lhs = y ; Rhs = vars,one ; Start = kx_0,b,s ; Maxit = 0 $ 
TOBIT ; Lhs = y ; Rhs = vars,one $ 
\end{verbatim}

Compute the likelihood ratio statistic.

\begin{verbatim}
CALC ; List ; lr = 2*(logl - l0) ; 1 - Chi(lr,kx) $ 
\end{verbatim}

This computes a Wald statistic.

\begin{verbatim}
MATRIX ; beta = b(1:kx) ; vb = varb(1:kx,1:kx) ; List ; Wald = beta’<vb>beta $ 
CALC ; List ; 1 - Chi(wald,kx) $ 
\end{verbatim}

Retained output from the model includes

Matrices: \( b, varb \)

Scalars: \( s \) = estimated \( \sigma \), \( ybar, sy, kreg \) = number of coefficients, \( nreg \) = number of observations, \( nonlimts \) = number of nonlimit observations in estimating sample
The diagnostic information for the model also includes Lin and Schmidt’s LM test for the model specification against the alternative suggested by Cragg as well as a test for nonnormality.

### 16.2.3 Marginal Effects

The marginal effects in the tobit model when censoring is at the left, at zero, are computed using

$$E[y|x] = \Phi(\beta'x/\sigma)[\beta'x + \sigma\phi(\beta'x/\sigma)/\Phi(\beta'x/\sigma)].$$

After some algebra, we find

$$\frac{\partial E[y|x]}{\partial x} = \Phi(\beta'x/\sigma)\beta.$$

The preceding is a broad result which carries over to more general models. That is,

$$\frac{\partial E[y|x]}{\partial x} = \text{Prob(nonlimit)}\beta$$

for all specifications of the censoring limits, whether in one tail or both. To obtain a display of the marginal effects for the tobit model, add

; Marginal Effects

to the TOBIT command. A full listing of the marginal effects computed at the sample means, including standard errors, the estimated conditional mean, and the scale factor, will be included in the model output.

### 16.3 Sample Selection Model

Many variants of the ‘sample selection’ model can be estimated with LIMDEP. Most of them share the following structure: A specified model, denoted A, applies to the underlying data. However, the observed data are not sampled randomly from this population. Rather, a related variable $z^*$ is such that an observation is drawn from A only when $z^*$ crosses some threshold. If the observed data are treated as having been randomly sampled from A instead of from the subpopulation of A associated with the ‘selected’ values of $z^*$, potentially serious biases result. The general solution to the selectivity problem relies upon an auxiliary model of the process generating $z^*$. Information about this process is incorporated in the estimation of A.

Several of the forms of this model which can be estimated with LIMDEP depart from Heckman’s now canonical form, a linear regression with a binary probit selection criterion model:

$$y = \beta'x + \varepsilon,$$

$$z^* = \alpha'w + u,$$

$$\varepsilon, u \sim N[0,0,\sigma_\varepsilon^2, \sigma_u^2, \rho].$$

A bivariate classical (seemingly unrelated) regressions model applies to the structural equations. The standard deviations are $\sigma_\varepsilon$ and $\sigma_u$, and the covariance is $\rho\sigma_\varepsilon\sigma_u$. If the data were randomly sampled from this bivariate population, the parameters could be estimated by least squares, or GLS combining the two equations. However, $z^*$ is not observed. Its observed counterpart is $z$, which is determined by
\[ z = 1 \text{ if } z^* > 0 \]
and
\[ z = 0 \text{ if } z^* \leq 0. \]

Values of \( y \) and \( x \) are only observed when \( z \) equals one. The essential feature of the model is that under the sampling rule, \( E[y|x,z=1] \) is not a linear regression in \( x \), or \( x \) and \( z \). The development below presents estimators for the class of essentially nonlinear models that emerge from this specification.

The basic command structure for the models described in this chapter is

\[
\text{PROBIT} ; \ Lhs = \text{variable } z ; \ Rhs = \text{variables in } w ; \ Hold \$
\]

\[
\text{SELECT} ; \ Lhs = \text{variable } y ; \ Rhs = \text{variables in } x \$
\]

Note that two commands are required for estimation of the sample selection model, one for each structural equation.

### 16.3.1 Regression Models with Sample Selection

The models described in this section are based on a dichotomous selection mechanism. Heckman’s approach to estimation is based on the following observations: In the selected sample,

\[
E[y_i | x_i, z_i = 1] = E[y_i | x_i, \alpha'w_i + u_i > 0] = \beta'x_i + E[u_i | u_i > -\alpha'w_i] = \beta'x_i + \rho\sigma_u \phi(-\alpha'w_i)/[1 - \Phi(-\alpha'w_i)] = \beta'x_i + \rho\sigma_u \phi(\alpha'w_i)/\Phi(\alpha'w_i).
\]

Given the structure of the model and the nature of the observed data, \( \sigma_u \) cannot be estimated, so it is normalized to 1.0. (We observe the same values of \( z_i \) regardless of the value of \( \sigma_u \).) Then,

\[
E[y_i | x_i, z_i = 1] = \beta'x_i + (\rho\sigma_u)\lambda_i = \beta'x_i + \theta\lambda_i.
\]

There are some subtle ambiguities in the received applications of this model. First, it is unclear whether the index function, \( \beta'x_i \), or the conditional mean is really the function of interest. If the model is to be used to analyze the behavior of the selected group, then it is the latter. If not, it is unclear. The index function would be of interest if attention were to be applied to the entire population, rather than those expected to be selected. This is application specific. Second, the marginal effects in this model are complicated as well. For the moment, assume that \( x_i \) and \( w_i \) are the same variables. Then,

\[
\frac{\partial E[y_i | x_i, z_i = 1]}{\partial x_i} = \beta + \theta(-\lambda_i\alpha'x_i - \lambda_i^2)\alpha.
\]
For any variable \(x_k\) which appears in both the selection equation (for \(z_i\)) and the regression equation, the marginal effect consists of both the direct part \((\beta_k)\) and the indirect part, which is of opposite sign – the term in parentheses is always negative; \(\theta(-\lambda_i\alpha'x_i - \lambda_i^2\alpha_k)\). It is not obvious which part will dominate. Most applications have at least some variables that appear in both equations, so this is an important consideration. Note also that variables which do not appear in the index function still affect the conditional mean function through their affect on the inverse Mills ratio (the ‘selection variable’). (We note the risk of conflict in the notation used here for the selection term, \(\lambda\), and the loglinear term in the conditional mean functions of the generalized linear models in the previous chapter. There is no relationship between the two. The two uses of ‘lambda’ are so common in the literature as to have become part of the common parlance and as such, the risk of ambiguity is worse if we try to change the notation used here for clarity.)

\(LIMDEP\) contains three estimators for this model, Heckman’s two step (or ‘Heckit’) estimator, full information maximum likelihood, and two step maximum likelihood (which is, more or less, a limited information maximum likelihood estimator). The two step estimator is given here. The others are documented in the full manual for the program.

### 16.3.2 Two Step Estimation of the Standard Model

Heckman’s two step, or ‘Heckit’ estimation method, is based on the method of moments. It is consistent, but not efficient estimator.

**Step 1.** Use a probit model for \(z_i\) to estimate \(\alpha\).

For each observation, compute \(\lambda_i = \phi(\alpha'w_i)/\Phi(\alpha'w_i)\) using the probit coefficients.

**Step 2.** Linearly regress \(y_i\) on \(x_i\) and \(\lambda_i\) to estimate \(\beta\) and \(\theta = \rho\sigma_e\).

Adjust the standard errors and the estimate of \(\sigma_e^2\), which is inconsistent.

The corrected asymptotic covariance matrix for the two step estimator, \((b,c)\), is

\[
\text{Asy.Var}[b,c] = \sigma_e^2 (X^*X^*)^{-1} \left[ X^* (I - \rho^2 \Delta) X^* + \rho^2 (X^* \Delta W) \Sigma (W^' \Delta X^*) (X^*X^*)^{-1} \right]
\]

where

- \(X^* = [X : \lambda]\),
- \(\Delta = \text{diag}[\delta]\),
- \(\delta_i = -\lambda_i (\alpha'w_i + \lambda_i)\) (-1 \(\leq \delta_i \leq 0\)),

and

\(\Sigma\) = asymptotic covariance matrix for the estimator of \(\alpha\).

A consistent estimator of \(\sigma_e^2\) is \(\delta_i^2 = e'e/n - \hat{\theta}^2\delta\). The remaining parameters are estimated using the least squares coefficients. The computations used in the estimation procedure are those discussed in Heckman (1979) and in Greene (1981).

**NOTE:** (This is one of our frequently asked questions.) \(LIMDEP\) always computes the corrected asymptotic covariance matrix, for all variants of selection models in all model frameworks.

The estimator of the correlation coefficient, \(\rho\), is \(\text{sign}(\hat{\theta}) \sqrt{\hat{\theta}^2/\delta_i^2}\). This is the ratio of a regression coefficient (the coefficient on \(\lambda_i\)) and the variance estimator above. Note that it is not a sample moment estimator of the correlation of two variables. This ratio is not guaranteed to be between -1 and +1. (See Greene (1981), which is about this result.) But, note also that an estimate of \(\rho\) is...
needed to compute the asymptotic covariance matrix above, so this is a potential complication. When this occurs, *LIMDEP* uses either +1 or -1, and continues. We emphasize, this is not an error, nor is it a program failure. It is a characteristic of the data. (It may signal some problems with the model.) When this condition occurs, the model results will contain the diagnostic

Estimated correlation is outside the range -1 < r < 1. Using 1.0

This condition is specific to the two step regression estimators. The maximum likelihood estimators discussed below force the coefficient to lie in the unit interval – Ρ is estimated directly, not by the method of moments.

To estimate this model with *LIMDEP*, it is necessary first to estimate the probit model, then request the selection model. The pair of commands is

```plaintext
PROBIT ; Lhs = name of z ; Rhs = list for w ; Hold results $
SELECT ; Lhs = name of y ; Rhs = list for x $
```

For this simplest case, ; Hold ... may be abbreviated to ; Hold. All of the earlier discussion for the probit model applies. (See Chapter E18.) This application differs only in the fact the ; Hold requests that the model specification and results be saved to be used later. Otherwise, they disappear with the next model command. The *PROBIT* command is exactly as described in Chapter E18. The selection model is completely self contained. You do not need to compute or save \( \lambda_i \).
<table>
<thead>
<tr>
<th>ANCILLARY PARAMETERS</th>
<th>8-16</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUTOCORRELATION</td>
<td>14-5, 14-9, 14-10</td>
</tr>
<tr>
<td>BINARY CHOICE</td>
<td>15-1</td>
</tr>
<tr>
<td>MARGINAL EFFECTS</td>
<td>15-6</td>
</tr>
<tr>
<td>MARGINAL EFFECT FOR DUMMY VARIABLE</td>
<td>15-7</td>
</tr>
<tr>
<td>PREDICTION</td>
<td>15-4</td>
</tr>
<tr>
<td>ROBUST COVARIANCE MATRIX</td>
<td>15-7</td>
</tr>
<tr>
<td>BINARY VARIABLES</td>
<td>7-10, 7-14</td>
</tr>
<tr>
<td>BOX-COX TRANSFORMATION</td>
<td>7-9</td>
</tr>
<tr>
<td>BROWSER</td>
<td>2-8</td>
</tr>
<tr>
<td>CALCULATE</td>
<td>6-23, 6-24, 10-1</td>
</tr>
<tr>
<td>ALGEBRA</td>
<td>10-6, 10-7</td>
</tr>
<tr>
<td>COMMANDS</td>
<td>10-2, 10-5, 10-6</td>
</tr>
<tr>
<td>CORRELATION</td>
<td>10-11, 10-13</td>
</tr>
<tr>
<td>FUNCTIONS</td>
<td>10-9, 10-10</td>
</tr>
<tr>
<td>MATRIX DIMENSIONS</td>
<td>10-10</td>
</tr>
<tr>
<td>MINIMUM AND MAXIMUM</td>
<td>10-11</td>
</tr>
<tr>
<td>NAMES</td>
<td>10-7</td>
</tr>
<tr>
<td>NORMAL CDF</td>
<td>10-10</td>
</tr>
<tr>
<td>REGRESSION STATISTICS</td>
<td>10-11</td>
</tr>
<tr>
<td>RESULTS</td>
<td>6-24, 10-4</td>
</tr>
<tr>
<td>R SQUARED</td>
<td>10-12</td>
</tr>
<tr>
<td>T DISTRIBUTION</td>
<td>10-10</td>
</tr>
<tr>
<td>CATEGORICAL VARIABLE</td>
<td>7-17</td>
</tr>
<tr>
<td>CENSORED REGRESSION</td>
<td>16-2</td>
</tr>
<tr>
<td>CENSORING</td>
<td>16-1</td>
</tr>
<tr>
<td>CHI-SQUARED TEST</td>
<td>6-25</td>
</tr>
<tr>
<td>CLUSTERING</td>
<td>15-8</td>
</tr>
<tr>
<td>COEFFICIENTS</td>
<td>8-14</td>
</tr>
<tr>
<td>COMMAND</td>
<td>4-1</td>
</tr>
<tr>
<td>COMMAND $ ENDING</td>
<td>4-2</td>
</tr>
<tr>
<td>2SLS</td>
<td>14-11</td>
</tr>
<tr>
<td>CALC</td>
<td>6-23, 6-24, 10-1</td>
</tr>
<tr>
<td>CREATE</td>
<td>6-13, 7-2, 7-9, 7-30</td>
</tr>
<tr>
<td>CROSSTAB</td>
<td>13-6</td>
</tr>
<tr>
<td>DATES</td>
<td>6-16, 7-24, 7-29</td>
</tr>
<tr>
<td>DELETE</td>
<td>7-21</td>
</tr>
<tr>
<td>DSTAT</td>
<td>13-1, 6-12, 6-17</td>
</tr>
<tr>
<td>ENDPROC</td>
<td>11-4, 11-5</td>
</tr>
<tr>
<td>EXECUTE</td>
<td>11-2, 11-4, 11-5</td>
</tr>
<tr>
<td>GO TO</td>
<td>11-5</td>
</tr>
<tr>
<td>HISTOGRAM</td>
<td>13-4</td>
</tr>
<tr>
<td>INCLUDE</td>
<td>7-24, 7-25</td>
</tr>
<tr>
<td>KERNEL</td>
<td>13-7, 13-8</td>
</tr>
<tr>
<td>LOAD</td>
<td>2-7</td>
</tr>
<tr>
<td>LOGIT</td>
<td>15-1, 15-2, 15-3</td>
</tr>
<tr>
<td>MATRIX</td>
<td>6-17, 6-23, 6-26, 9-1, 9-4</td>
</tr>
<tr>
<td>NAMELIST</td>
<td>6-17, 6-22, 6-26, 7-21, 7-22</td>
</tr>
<tr>
<td>OPEN</td>
<td>2-7</td>
</tr>
<tr>
<td>ORDERED CHOICE</td>
<td>15-9</td>
</tr>
<tr>
<td>PERIOD</td>
<td>6-16, 7-24, 7-28, 7-29</td>
</tr>
<tr>
<td>PLOT</td>
<td>6-16, 13-10, 13-12</td>
</tr>
<tr>
<td>PROBIT</td>
<td>15-1, 15-2, 16-5</td>
</tr>
<tr>
<td>PROC</td>
<td>10-2, 11-4, 11-5</td>
</tr>
<tr>
<td>READ</td>
<td>6-4, 6-6, 7-1, 7-7</td>
</tr>
<tr>
<td>REGRESS</td>
<td>3-11, 3-12, 6-19, 6-22, 6-25, 6-26, 7-2, 14-1, 14-4</td>
</tr>
<tr>
<td>REJECT</td>
<td>7-24, 7-25</td>
</tr>
<tr>
<td>SAMPLE</td>
<td>7-24, 7-25</td>
</tr>
<tr>
<td>SAVE</td>
<td>2-7</td>
</tr>
<tr>
<td>SELECTION</td>
<td>16-4, 16-5</td>
</tr>
<tr>
<td>SPLOT</td>
<td>7-21</td>
</tr>
<tr>
<td>TOBIT</td>
<td>16-1</td>
</tr>
<tr>
<td>WALD</td>
<td>6-26</td>
</tr>
<tr>
<td>COMMAND BUILDERS</td>
<td>4-4, 4-5, 6-17, 8-3, 8-4, 8-5, 12-3</td>
</tr>
<tr>
<td>COMMANDS</td>
<td>3-7, 3-9, 4-1</td>
</tr>
<tr>
<td>COMMAND FILE</td>
<td>2-6, 6-5</td>
</tr>
<tr>
<td>COMMAND, MODEL</td>
<td>8-1, 8-10, 12-1</td>
</tr>
<tr>
<td>LEFT HAND SIDE VARIABLE</td>
<td>12-2</td>
</tr>
<tr>
<td>OUTPUT</td>
<td>12-5</td>
</tr>
<tr>
<td>RIGHT HAND SIDE VARIABLES</td>
<td>12-2</td>
</tr>
<tr>
<td>ROBUST COVARIANCE MATRIX</td>
<td>12-5</td>
</tr>
<tr>
<td>VARIABLES</td>
<td>12-2</td>
</tr>
<tr>
<td>CONSTANT ELASTICITY</td>
<td>6-1</td>
</tr>
<tr>
<td>CONSTANT TERM</td>
<td>6-19, 8-2</td>
</tr>
<tr>
<td>COPY/PASTE</td>
<td>6-6</td>
</tr>
<tr>
<td>CORRELATION</td>
<td>6-17, 10-13</td>
</tr>
<tr>
<td>COVARIANCE MATRIX</td>
<td>8-8, 8-15, 8-16, 14-5</td>
</tr>
<tr>
<td>CREATE</td>
<td>6-13, 7-9, 7-10, 7-11 to 7-13, 7-30, 7-31</td>
</tr>
<tr>
<td>FUNCTIONS</td>
<td>7-14</td>
</tr>
<tr>
<td>CREATE, CATEGORICAL VARIABLE</td>
<td>7-17</td>
</tr>
<tr>
<td>CROSS SECTION</td>
<td>7-25</td>
</tr>
<tr>
<td>CROSSTABULATION</td>
<td>13-6</td>
</tr>
<tr>
<td>LABELS</td>
<td>13-6</td>
</tr>
<tr>
<td>WEIGHTS</td>
<td>13-7</td>
</tr>
<tr>
<td>CURRENT SAMPLE</td>
<td>7-24, 7-25, 7-26</td>
</tr>
</tbody>
</table>
Index

Data area 3-1, 7-1
Data editor 6-8, 7-1, 7-2
Data file, variable names 7-6
Data matrix 9-8, 9-9
Data, missing 7-8, 7-30, 7-31
Data, read 6-6
Data, time series 6-16
Descriptive statistics 6-12, 13-1
  Missing values 13-2
  Quantiles 13-2
  Weights 13-2
Desktop 2-3
Dialog boxes 4-4
Discrete choice models 15-1
  Chi squared 15-2
  Hypothesis test 15-2
  Log likelihood 15-2
Dummy variable, partial effect 15-7
Edit/Copy/Paste 3-7
Editing window 3-2, 3-4, 3-6, 3-9
Elasticity 6-10
Enter key 3-7
Entropy 15-4
Excel 6-4, 7-8
  File format 6-4
Exit program 2-5
Exit status 3-11
F statistic 6-23, 6-24, 6-26
  CALC 6-26
File, ascii 7-6
File, data 7-3
File, input 6-5
File menu 2-8, 3-2, 3-3, 6-15
  File:New 3-3
Filenames 2-6, 7-5
Font 3-5
Gasoline market 6-2, 6-3
Go button 3-6, 3-9, 3-11, 6-5, 6-9
Graphics 5-2
  Export 5-2
Hausman test 14-17, 14-23
Help 3-14
  Windows Vista 3-14
Heteroscedasticity 1-1, 8-11, 14-9
Histogram 13-4
  Continuous data 13-5
  Discrete data 13-6
Hypothesis test 6-25, 6-26
  t ratio 6-27
  Structural change 6-28, 6-29
Hypothesis test 14-8
Icon 2-3
INCLUDE command 7-24, 7-26
Information criteria 15-3
Input file 6-4
Insert in text window 3-3
Installation 2-1
Instrumental variables 14-11
Kernel density 13-7, 13-8
Kurtosis 13-1
Labels, observation 7-7
Lagged values 7-11, 7-15
LM test 14-16, 14-23
Least squares 6-23
Likelihood ratio test 10-1
LIM file 3-3
Limited dependent variable 1-1
Lists of variables 6-17
LOAD command 2-7
Logistic 7-14
Logit 15-1, 15-2, 15-3
  Ordered 15-9
Logs, data 6-1, 6-13, 7-11, 7-14
Log likelihood 8-14
Looping, program 11-4, 11-5
Matrix 7-16, 9-1
  Algebra 9-13
  Calculator 9-3
  Characteristic roots 9-7, 9-18
  Command builder 9-4
  Creating 9-16
  Data 9-8, 9-9
  Defining 9-14
  Determinant 9-19
  Diagonal 9-17
  Expression 9-13
  Identity 9-17
  Namelists defining 9-10
  Names conventions 9-11
  Partitioned 9-16
  Product 9-13
  Results 9-5, 9-6
  Statistical output 9-7
  Sums 9-10, 9-11, 9-19
Index

Trace 9-19
Transpose 9-13
Windows 9-5
MATRIX command 6-17, 6-23
Matrix algebra 9-1, 9-13
Maximum 7-9
Mean vector, matrix 9-20, 9-21
Missing data 7-30
Missing values 7-8, 7-12, 7-13
Model command 4-1
Models 1-1, 8-1, 12-1
Model command 8-10
Model, constant term 8-2
Model types 12-3, 12-4, 12-5
Moment matrix 9-11, 9-20
Mouse button 3-8

Namelists 7-21, 7-22, 7-23
NAMELIST 6-17, 6-25
Names 4-1
Conventions 4-2
Reserved 4-2, 4-3
Syntax 4-1
Newey-West 14-5
Nonlinear models 1-1
NormalCDF 7-14
Normal distribution 7-14

OLS results 8-12
OPEN command 2-7, 3-1
Options window 2-4
Ordered choice 15-9
Output 15-11
Marginal effects 15-13
Output 4-6, 5-1, 5-2
Editing 5-2
Exporting 5-2
Output file 2-6
Output, model 8-7, 8-12, 8-13
Output window 3-10
Output window 8-6

P values 8-14
Pagan, A. 1-2
Panel data 14-12
Balanced panel 14-12
Count variable 14-14, 14-15
Data arrangement 14-13
Fixed effects 14-15, 14-18
Hausman test 14-17, 14-23
LM test 14-16, 14-23

R squared 14-24
Random effects 14-15, 14-21
Results 14-19
Stratification 14-13
Path, file 7-5
PERIOD command 7-28, 7-29
Plot 13-10
Graphics file 13-10
Multiple scatter plots 13-14
Regression 13-13
Scaling 13-15
Scatter plot 13-10, 13-12
Time series 13-13
PLOT command 6-16
Title 6-16
Time series 6-16
Plot data 6-16
Plot regression residuals 6-21, 6-23
Poisson model 8-12, 8-19
Predictions 8-16, 12-6
Missing values 8-16, 8-17
Probit 10-1, 15-1, 15-2
Ordered 15-9
Procedure
Parameters 11-7, 11-8
Silent 11-6
Programming 1-2
Programming 11-1, 11-4
Project 2-4, 3-1, 6-14
Project window 3-4, 3-12, 3-13, 4-3, 6-13, 6-24, 8-4, 8-15

R squared 6-24, 8-15
Random number generator 7-19
Seed 7-20
Random samples 7-19
Discrete 7-20
Read ASCII data 6-6
Command window 6-6
READ command 6-3, 6-6, 7-1, 7-7
Spreadsheet 7-7
Reading data 7-3
Registration 2-1, 2-2
REGRESS 8-2, 8-3, 8-5
REGRESS command 3-11, 3-12, 6-22
Regression 3-11, 6-1, 6-12, 6-19, 14-1
Autocorrelation 14-9, 14-10
Constant term 14-1
Fixed effects 14-15, 14-18
Heteroscedasticity 14-9
Hypothesis test 14-8
Left out variable 6-20
Model 14-1
Multiple 6-22
Panel data 14-12
Plot residuals 14-4
Predictions 14-4
Random effects 14-15, 14-21
Residuals 14-3
Residual plot 6-23
Results 14-2, 14-3
Restricted 14-5, 14-6
Retrievable results 14-3
Robust covariance matrix 14-5
Regression, constant term 6-19
Reserved names 4-3
REJECT command 7-24, 7-26
Residuals, plot 6-21
Residuals 8-16
Plot 8-18
Residuals 12-6
Restricted least squares 9-1, 14-6
Restricted regression 6-27
MATRIX 6-27
t statistic 6-28
Retrievable results 8-14
Rhs list 8-1
Robust covariance matrix 12-5, 14-5
2sls 14-11
Panel data 14-21
Run menu 3-6, 3-7
Sample 7-24
SAMPLE command 7-25, 7-27
Sample observations 7-3
Sample selection 16-1, 16-4
Heckman estimator 16-6
Model 16-5
PROBIT 16-5
Selection equation 16-5
Two step estimation 16-6
Sandwich estimator 15-7
SAS 2-7
Save 2-5
SAVE command 2-7
Scalars 6-24, 8-15
Seed, RNG 7-20
Session 3-1
Skewness 13-1
SKIP 7-31
Spreadsheet 6-4
Spreadsheet, importing, 7-7

MATRX 6-29
Chi squared 6-29
Standard errors 8-14
Stat Transfer 2-7
Stata 2-7
Stratification 14-13
Structural break 6-29
Sums of observations, matrix 9-19
Weighted 9-20
Syntax, command 4-1

Text editing 3-3, 3-5
Text editor 11-1
Executing commands 11-2
Textbooks 1-2
Time series 6-16
Time series data 7-28
Tobit 16-1
Hypothesis tests 16-3
Marginal effects 16-4
Results 16-3
Toolbar 3-6, 7-1
Trace 2-5
Trace file 3-11
Transformations, data 6-1, 6-13, 7-9, 7-30
Trend variable 7-13, 7-14
Tutorial, commands 6-10, 6-11
Two stage least squares 14-11
Results 14-11

Variable list 7-21
Variable names 7-6
Variables, new 7-2
Variables, transformed 7-9
Vella, F. 1-2
Vista 3-14

Wald statistic 6-26, 6-29
Weights 8-10
Wts 8-10
Heteroscedasticity 8-11
Scaling 8-11

Window
Editing 3-3
Output, 3-10, 4-6, 5-1, 5-2
Project 3-13, 4-3, 6-13
Window, editing 3-2
Word processor 3-4

XLS file 6-4