TSP 5.0 User's Guide

Bronwyn H. Hall

and

Clint Cummins

First edition (Version 4.0) published 1980.

TSP is a software product of TSP International. The information in this document is subject to change without notice. TSP International assumes no responsibility for any errors that may appear in this document or in TSP. The software described in this document is protected by copyright. Copying of software for the use of anyone other than the original purchaser is a violation of federal law. Time Series Processor and TSP are trademarks of TSP International.

For further information, contact:

TSP International
PO Box 61015
Palo Alto, CA 94306, USA
phone: (650) 326-1927
fax: (650) 493-2912
http://www.tspintl.com/
email: info@tspintl.com

Table of Contents

Table of Figures

Preface

The User's Guide for TSP had its origins in the User's Guide to Version 3.5, co-authored by Bronwyn Hall and Robert Hall in 1980. Over the years, it has been completely revised as the TSP program has been enhanced and expanded, and the current edition represents a major revision. Contributions to earlier versions of the manual were also made by Rebecca Schnake and editorial contributions by Chris Hall, Lila Havens, Araya Niemloy, Margaret Reeves, and Rossannah Reeves. We are very grateful to all these people for their efforts.

This version is the first to be printed a new, more compact and convenient format, and we hope you will find it useful. We have tried wherever we can to document the sources of our procedures and algorithms in the scientific literature, and you will find a comprehensive set of references to this literature at the end of the volume.

Bronwyn H. Hall
Clint Cummins
February 2005

1. INTRODUCTION

TSP is a general-purpose computer language for econometric and statistical data processing and estimation. The program can be used for any of the following tasks:

Applied econometrics, including teaching
Macroeconomic research and forecasting
Sales forecasting
Financial analysis
Cost analysis and forecasting
Monte Carlo simulation
Estimation and simulation of economic models

Currently, TSP is installed on thousands of computers worldwide -- from small stand-alone personal computers to large mainframes and shared systems. Although TSP was originally and continues to be developed primarily by economists, there is nothing in its design limiting it to economic time series. Any data consisting of repeated observations of the same variable for different units may be analyzed with TSP.

The basic data object within TSP is the series. Each series has a name, and you can request operations on all the observations just by mentioning the name of the series. TSP provides convenient ways to enter series, to create new series from existing ones, to display and print series, and to carry out statistical analysis of the relations among series.

Some of the most important TSP features are:

- Both data and commands are entered in free format.

- Data can be transformed by convenient algebraic statements.

- Leads and lags are specified in a natural way.

- There are few restrictions on the order of the operations in a run.

- The output from one statistical procedure can easily be used as the input to another.

All standard econometric techniques are available in an accurate and efficient form: ordinary least squares, two-stage least squares (instrumental variables), limited information maximum likelihood (LIML), polynomial and Shiller distributed lags, autoregressive correction, and weighted least squares.

Advanced techniques are available, including nonlinear least squares, estimation of GARCH models, Box-Jenkins estimation, multivariate regression, three stage least squares, GMM, full information maximum likelihood, estimation with qualitative dependent variables, Kalman filter estimation, programmable maximum likelihood, and solution of linear and nonlinear models.

A full set of matrix operations and the analytic differentiation procedures makes it possible for you to program your own estimators easily.

Panel and cross-section data sets can be handled by TSP as easily as time series. (Some users have gone as far as 1 million observations or more.)

TSP is a large software system and econometric language with many features, and it is not possible or desirable to learn them all at once. This manual is intended to provide an introduction to TSP and its most commonly used features. It does not describe all of the procedures, nor indicate the full power of the program as a language. For detailed information on each TSP command and complete references to the econometric and statistical literature, consult the TSP Reference Manual, available where you obtained this manual or from the address at the front of this manual. For more complex examples and solutions for special problems, see the TSP examples page on the TSP International web site (http://www.tspintl.com). This web site also documents the very latest features of the programs, as well as any known bugs and work arounds.

You may find it helpful to know a little about how this guide is organized. The guide is divided into 4 major sections plus appendices:

Section I: Chapters 1 - 4

Section I covers the basics of TSP. Chapter 2 proceeds on the assumption that you are a complete TSP novice. We introduce some of the most basic ideas of TSP and show you the input and the output for a simple but complete TSP job. Chapter 3 covers the fundamental concepts of TSP in a more thorough way and concludes with a more elaborate example. This example is also included on the installation CD, under the name ***illus.tsp***.

If you are using this program on a personal computer, there are two different ways to work: with a batch input file that you edit before running the program, or in an interactive mode. Chapter 4 introduces the basics of working in interactive mode.

Beginning with TSP Version 4.5, it is also possible to use TSP within Givewin, a Windows-interface utility that provides full graphics support for TSP (including editing and printing) as well as a data and program editing facility.[1] Givewin allows both interactive and batch use of TSP. You can read about using Givewin with TSP in Appendix C.[2]

Section II: Chapters 5 - 6

Section II, entitled Linear Estimation, describes the most commonly used features of the program, such as linear regression models, and printing and plotting the data. Methods discussed include ordinary least squares, two stage least squares (instrumental variables), and obtaining simple statistics on your variables.

Section III: Chapters 7 – 10

In Section III, we present the powerful and extensive range of tools in TSP for nonlinear estimation. Methods covered include nonlinear single and multi-equation least squares, general maximum likelihood, GMM, and qualitative dependent variable models (Probit, Multinomial Logit, Tobit, and sample selection). TSP's nonlinear estimation methods are generally iterative and make use of program-generated analytic gradients of the objective function, which are more accurate and faster than numeric gradients. Section III also discusses hypothesis testing and TSP's flexible equation manipulation system.

[1] Givewin is authored by Jurgen Doornik and David Hendry and distributed by Timberlake Consulting (http://www.timberlake.co.uk). It is distributed with TSP at an additional charge. Other econometrics programs than can be used within the Givewin environment include PCGive, PCFIML, STAMP, and the Ox language. See the Givewin website at http://www.nuff.ox.ac.uk/Users/Doornik/ for further information on these packages.

[2] Prior to TSP version 4.5, TSP through the Looking Glass (TLG), provided an interface similar to the Givewin interface with all the standard Windows file handling and editing features. Although this interface is still available, Givewin is preferred because of its better graphics support.

Section IV: Chapters 11 – 16

Section IV discusses a series of more specialized and advanced procedures. These include Box-Jenkins (ARIMA) procedures, GARCH estimation, the Kalman filter, VAR estimation, cointegration testing, forecasting and model simulation, matrix computations and controlling the order of execution of your program. This section also treats the storage, reading, and writing of large amounts of data.

Section V: Appendices

The first few appendices give the basic syntax rules of TSP, and a list of new features in Version 5.0. These are followed by specific instructions for using TSP on various platforms, such as DOS/Window PCs, Apple Macintosh, and unix.

2. GETTING STARTED

2.1 Tools you will need to use TSP

The hardware you will need to use TSP depends on whether you want to work on your own personal computer or on a shared larger computer. One advantage of TSP is that since essentially the same program runs on both types of computers, you can develop your program on a smaller, cheaper personal computer, and then move to a larger computer to take advantage of its greater speed and data storage. TSP input files (***.tsp*** files) are text files that are easily uploaded (transferred to another computer) over a network connection.[3]

In addition to TSP, you will need one other piece of software to work efficiently with the program: a text editor for preparing input and reading output. If you run TSP under Windows, you can use *Givewin* to edit and run your batch files, and examine the results.[4] Contact your TSP supplier or TSP International if you wish to order this program and failed to do so in your original TSP order.

In this chapter, we assume that you know which computer you want to use and have chosen the appropriate version of TSP for that computer. First we discuss how to use the program on personal and networked computers. Then we give a simple introductory example of a TSP program to help you get started.

2.1.1 On a personal computer

TSP runs on most personal computers that run Windows, DOS, Linux or Mac OS. It also runs on some other unix systems.

You will probably find it convenient at first to work in interactive mode, as described in Chapter 4. Interactive mode is good for trying things out, and for quick one-time computations. Later, if you find yourself repeatedly typing the same commands, you can use a separate text editor (such as *Givewin*, the DOS EDIT command, Wordpad, or a word processing program) to edit your commands and save them in a text file. If you use a word-processor to prepare

[3] Consult your local system consultant for information on how to upload if you are unfamiliar with how to do it. Usually *ftp* is the method to use.

[4] Versions of TSP prior to Version 5.0 offered another Windows interface as well, *TSP through the Looking Glass* (TLG). The Givewin interface is preferred due to its greater support for graphics.

TSP programs, be sure to save the file as a DOS text or ASCII file. You can submit your commands (program) to TSP as an INPUT file to be executed. In *Givewin*, this is done simply by clicking on the “run” icon.[5] Setting up a session with an INPUT file and then using interactive mode to try new ideas is often the most efficient way to work.

2.1.2 On a shared or network computer

We assume that you or someone else has successfully installed TSP. Separate documents available from TSP International provide instructions for installation. The usual shared or networked installation is either a unix system (see Appendix E), or a networked Windows installation (see Appendix C).

2.2 A little vocabulary

Very little knowledge of computers is required to use TSP, but you will need to know the following computer terms to understand this manual:

Memory. A computer has millions of characters of high-speed memory. As your TSP job runs, all your data are in your computer's memory so that they can be reached quickly. At the conclusion of the run, the computer forgets everything in memory.

Disk. To save information between runs, the computer has space to store millions of characters on magnetic disks. The time required to find something on disk is several orders of magnitude longer than the time for memory, so anything that will be used intensively is moved from disk to memory at the beginning of a run.

File. Information on disk is stored in "files". TSP uses *input files* containing commands and data (***.tsp***) , *plain data files* for use by different input files (usually ***.txt***, ***.dat*** or ***.raw***), and *output files* containing your results (***.out***). As indicated, normally TSP input files have the extension ***tsp*** and TSP output files have the extension ***out***. Data files may have any extension. TSP databanks have the extension ***.tlb***. Spreadsheet data files use ***.xls*** and ***.wks*** (and some variations of those extensions).

Some basic TSP concepts that you will need to know are:

Series. A series is a set of observations on a variable, usually evenly spaced

[5] This icon is a tiny running man, shown on the main *Givewin* toolbar.

over time. Annual observations on GNP are an example we will use frequently. Monthly sales of a firm are another example. Cross-sectional, survey, and panel data are also referred to as series.

Command (or Statement). To communicate your wishes to TSP, you use commands. A command has a short, easy to remember name such as PRINT or OLSQ, followed in most cases by a list of series.

Program. You assemble a group of commands into a program, and then ask TSP to execute the program all at once. A program can consist of just a few commands, each to be executed once, or a large number of commands and a sophisticated structure for repeating some of them.

Input file. The input file contains your TSP program and the data required to run it, or instructions as to where to find the data.

Output file. TSP creates an output file as it executes your program. The file contains messages describing what TSP thinks are errors in your program or data, memos to you about what TSP has done in the course of executing your program, and your regression and other substantive results. When your computer has finished the TSP run, you should examine the output file with your text editor. If you are using *Givewin*, this is easily done because the output window appears on the screen as the program runs.

2.3 A simple regression example

In the rest of this chapter, we present a simple example of running a regression in TSP. The function of the commands are described briefly to give you a feel for using TSP. Using this very simple TSP program as an example, you should be able to set up a regression run of your own. The output for the program is shown in **Example 2.1**. To learn more about the basics of TSP, continue on to Chapter 3.

In the example run, we read two series of annual time series data (a sales variable and Gross National Product (GNP) for 1976 to 1985), take their natural logarithms, compute their means and standard deviations, and run a simple linear regression of the log of sales on the log of GNP. Here is the example:

```
OPTIONS CRT;   ? simple example
FREQ A;  SMPL 76,85;
LOAD SALES,GNP;
11.7 1706
13.7 1901
11.4 2151
12.3 2391
19.4 2608
20.4 2956
18.2 3051
25.3 3261
24.3 3639
28.3 3843
LSALES = LOG(SALES);  LGNP = LOG(GNP);
MSD LSALES,LGNP;   OLSQ LSALES C,LGNP;
```

Note that commands finish with a semicolon (;), which allows there to be more than one command per line or continuation of a command across several lines. If you are using TSP interactively, the semicolon at the end of each command is not required, and the \ key is used to continue a long command to the next line:

```
LOAD  LABOR LABOR2 CAPITAL CAPITAL2 MATERIAL \
MATER2
```

If you are using TSP in batch mode, the semicolon at the end of each command *is* required, and the \ key **cannot be** used:

```
LOAD  LABOR LABOR2 CAPITAL CAPITAL2 MATERIAL
MATER2 ;
```

The function of each command in the simple TSP program above is as follows:

OPTIONS CRT;	Keeps the output file width< 80 columns
? simple example	A comment that is ignored by TSP
FREQ A;	Specifies annual frequency for the data
SMPL 76,85;	Sets the range of data - from 1976 to 1985
LOAD SALES,GNP;	Reads two series using the nos. that follow
LSALES = LOG(SALES);	Creates a new variable = log SALES
MSD LSALES,LGNP;	Computes means & other statistics
OLSQ LSALES C,LGNP;	Regresses LSALES on LGNP and a constant

```
                              TSP Version 5.0
                       ( 1/17/05) TSP/GiveWin  4MB
                  Copyright (C) 2005 TSP International
                           ALL RIGHTS RESERVED
                            02/12/05 11:55AM
            In case of questions or problems, see your local TSP
            consultant or send a description of the problem and the
            associated TSP output to:
                            TSP International
                             P.O. Box 61015
                           Palo Alto, CA 94306
                                   USA
         PROGRAM
COMMAND  ***************************************************************
1  OPTIONS CRT;  ? simple example
2  FREQ A; SMPL 76,85;
4  LOAD SALES,GNP;
4  11.7 1706
4  13.7 1901
4  11.4 2151
4  12.3 2391
4  19.4 2608
4  20.4 2956
4  18.2 3051
4  25.3 3261
4  24.3 3639
4  28.3 3843
4  LSALES = LOG(SALES); LGNP = LOG(GNP);
7  MSD LSALES,LGNP;
8  OLSQ LSALES C,LGNP;
         EXECUTION
*******************************************************************************

Current sample:  1976 to 1985

                             Univariate statistics
                             =====================

Number of Observations: 10

                  Mean       Std Dev       Minimum       Maximum
LSALES         2.86666       0.33996       2.43361       3.34286
LGNP           7.88696       0.27326       7.44191       8.25401


                   Sum      Variance      Skewness      Kurtosis
LSALES        28.66657       0.11558     -0.053501      -1.67063
LGNP          78.86961      0.074670      -0.30857      -1.04519

                                     Equation   1
                                     ============

                      Method of estimation = Ordinary Least Squares

Dependent variable: LSALES
Current sample:  1976 to 1985
Number of observations:  10

       Mean of dep. var. = 2.86666      LM het. test = 1.86919 [.172]
  Std. dev. of dep. var. = .339965     Durbin-Watson = 2.13492 [.424,.744]
Sum of squared residuals = .171653  Jarque-Bera test = 1.15536 [.561]
   Variance of residuals = .021457   Ramsey's RESET2 = 1.38103 [.278]
Std. error of regression = .146481   F (zero slopes) = 40.4785 [.000]
               R-squared = .834978    Schwarz B.I.C. = -3.83236
      Adjusted R-squared = .814351    Log likelihood = 6.13495

           Estimated    Standard
Variable  Coefficient     Error       t-statistic   P-value
C         -6.09956      1.41004       -4.32581      [.003]
LGNP      1.13684       .178685       6.36227       [.000]

*******************************************************************************
END OF OUTPUT.
```

Figure 2.1 Simple TSP Run

3. TSP FUNDAMENTALS

This chapter introduces some basic TSP concepts and describes commands for setting up the sample of observations, reading data, making transformations, and setting options. It concludes with a fairly large example illustrating these concepts and some estimation methods, ranging from OLS to FIML.

3.1 Describing the sample of observations: FREQ, SMPL

Before beginning a TSP program, you need to specify the frequency of your data and the number of observations with the FREQ and SMPL commands. The FREQ command tells TSP how often a year each series is observed. For example,

```
FREQ A ;
```

specifies annual data, observed once a year. Other frequencies TSP understands are

Q	Quarterly (four times per year)
M	Monthly (12 times per year)
W	Weekly (52 times per year)
num	Any number less than 32,768 (example: 26 times per year)
N	Undated (usually cross-sectional or survey data)

If no frequency is specified, TSP assumes the frequency **N** (none).

A special FREQ statement is available for panel data (time series-cross section data). For details, see Chapter 15, on using panel data in TSP.

The **SMPL** command defines the set of observations to be used. For example, in the simple run in the last chapter,

```
SMPL 76,85;
```

informed TSP that observations dated from 1976 through 1985 were to be used. You only need to use a SMPL command when you want to change the observations you are using. There must be a sample in effect before the first operation on a series, but if you read data from a file without specifying a SMPL, the sample is defined for you, based on the contents of the file.

Formats for dates (the SMPL endpoints) are:

Annual: the full year or the last digits of the year (for the twentieth and twenty-first centuries). Examples: 2005, 1981, 1895, 82 (1982), 101 (2001). 200 would normally denote the year 2100 (using the default OPTIONS BASEYEAR=1900), while 201 would be just the year 201.[6] Negative years are valid; they are "B.C.".

Quarterly, Monthly, Weekly, and *numeric*: the full year or the last two digits of the year, colon, and the quarter, month, week, or period number. Examples: 1972:1, 65:4, 80:11. Obviously, quarters beyond 4 and months beyond 12 are invalid, as are negative or absent values.

No Frequency: use the observation numbers. Examples: 1, 2000.

More than one pair of dates may appear in a SMPL command. For example,

```
SMPL 59,70 72,83;
```

omits the year 1971. To omit several scattered observations, use SELECT or SMPLIF (see Section 3.3).

The dates in a SMPL statement must be in ascending order -- SMPL 70,59; would be invalid.

3.2 *Reading data into TSP: READ*

There are several ways to input your data into TSP. You can include it with the program or put it in a separate file. The separate file can be in a variety of formats, including spreadsheet and *MicroTSP, Eviews,* or *stata*. This chapter describes the simplest and easiest methods, reading data within the TSP program and reading data from an external file. See Chapter 16 for more complex examples.

3.2.1 Reading data in free format within the program

If you have only a few short series, you will probably find it convenient to read the data in free format within the program. To read data in free format, use the

[6] Therefore, TSP is year-2000 compliant, and can handle dates either before or after 2000.

READ command followed by the names of one or more series (names can be up to 64 characters long). The data follows immediately in free format, observation by observation, with one column per series. The numbers are separated from each other by blanks or commas. Here is the example from Chapter 2:

```
FREQ A; SMPL 76,85;
READ SALES,GNP;
11.7 1706
13.7 1901
11.4 2151
12.3 2391
19.4 2608
20.4 2956
18.2 3051
25.3 3261
24.3 3639
28.3 3843
```

The data for SALES and GNP could also be read separately (by rows) if it was more convenient. Semicolons after the data are optional:

```
FREQ A; SMPL 76,85;
LOAD SALES;
11.7 13.7 11.4 12.3 19.4 20.4 18.2 25.3 24.3 28.3 ;
LOAD GNP;
1706 1901 2151 2391 2608 2956 3051 3261 3639 3843 ;
```

If the number of data points does not correspond to the current sample, an error message will note the discrepancy. Missing values can be entered as a period (.), or the SMPL can be set to read in only those observations that are available. For example, if you had data on imports (IMPT) only for 1977 and later in the above example, use

```
SMPL 77,85;
LOAD IMPT;
35 42 67 85 90 92 97 120 126 ;
```

If you have a large volume of data, the commands and data can be moved to the bottom of the program, after an END; statement which terminates the data analysis. A NOPRINT; command after the END; statement will suppress printing of the commands and data in the output file. This is recommended after an initial run has checked for any error messages in loading the data. See Section 3.6 for more details.

Matrices may also be loaded -- see READ in the *Reference Manual* or Section 12.2.1 in this manual for more information on loading and using matrices. The matrix type and dimensions are required.

3.2.2 Reading data from an external file

For large datasets, you will probably read the data from a separate file created with a text editor, written by another program, or copied from disk or the web. To input the file, use the READ command with the FILE='filename' *option*. (Note: TSP *Options* are always enclosed in parentheses following the command name.) The following example would read a file containing just the data from the example in Chapter 2:

```
READ(FILE='SALESGNP.DAT') SALES,GNP;
```

The file SALESGNP.DAT could look like the following (free format allows more than one observation per line, or one observation split across more than one line):

```
11.7 1706
13.7 1901
11.4
2151
12.3 2391 19.4 2608 20.4 2956
18.2 3051 25.3 3261 24.3 3639 28.3 3843
```

The next example would read the Excel file, SML.XLS. For information on reading spreadsheet files, see Chapter 16.

```
READ(FILE='SML.XLS') ;
```

3.3 Selection of observation subsets: SELECT, SMPLIF

In addition to the SMPL command, the sample can also be selected by rules based on the values of series using SELECT and SMPLIF commands. The difference between SELECT and SMPLIF is that successive SMPLIFs nest (define smaller and smaller subsets of observations) while successive SELECT statements do not. For example, suppose you want to run a regression on only the positive values of the variable X, but these values are scattered throughout a dataset with 200 observations. It would be tedious to specify all the gaps for omitted observations in a SMPL statement. Instead, use SELECT:

```
SMPL 1,200;
SELECT X>0;
OLSQ Y C,X;
```

SELECT always refers to the most recent SMPL statement to define the overall sample of observations -- any intervening SELECT or SMPLIF statements are ignored. As a result, successive SELECT statements do not "nest" within each other. Nesting is provided with the SMPLIF command, which includes only those observations from the *current sample* which make the selection criteria true. For example,

```
SMPL 1 200 ;
SMPLIF X>0 ;
OLSQ Y C X ;
SMPLIF X<=10 ;
OLSQ Y C X ;
```

is equivalent to

```
SMPL 1 200 ;
SELECT X>0 ;
OLSQ Y C X ;
SELECT X>0 & X<=10 ;
OLSQ Y C X ;
```

Note that the samples defined by SELECT and SMPLIF remain in effect until the next SMPL, SELECT, or SMPLIF statement. The "full sample" in these examples could be restored in one of two ways:

```
SMPL 1,200;          or          SELECT 1;
```

(because the value 1 makes the selection criterion true for all observations in the previous SMPL statement).

In either a SMPLIF or a SELECT statement, the selection criterion can be any expression like those used in GENR (see Section 3.5). When the expression is evaluated for each observation, values that are positive will be treated as true and values that are zero or negative as false.

3.4 Missing Values

TSP supports missing values in data series. In free format input files you should indicate missing data by a period (.); in fixed format files you will have to use another value and then recode that value to the missing value code (see Section 3.5.1 for an example). TSP names the missing value code @MISS so that you can refer to it in your program.

The WRITE statement also uses a period (.) to indicate missing values when printing or writing output files. Procedures that transform data such as GENR automatically propagate missing values for most operation when they are present (missing values are also generated if there are arithmetic errors in the transformations).

Many statistical procedures (MSD, OLSQ, INST, AR1) automatically delete observations with missing values before executing. Others (the time series procedures such as ARCH) print an error message if you attempt estimation with missing data. A complete list of the procedures that cannot estimate gaps in the data is given in the *Reference Manual*. You can use SELECT with the MISS() function to skip missing observations:

```
SELECT .NOT. MISS(SALES) .AND. .NOT. MISS(GNP) ;
```

will choose only observations for which SALES and GNP are not missing for the estimation procedure.

3.5 Creating new series with transformations: GENR

The GENR command creates a new series based on a formula you supply. For example,

```
GENR Z=LOG(REV);
```

forms a new series called Z, the natural logarithm of REV. Series Z is defined only for those observations in the current sample; any other observations of Z that are used later will have a missing value code as their value. A missing value code is also given to any observation that has an arithmetic error when computed. For example, if one of the observations of REV were zero, Z would be missing for that observation since the log of zero is undefined. If Z is used later in another GENR, the missing value will propagate, that is, the new series will also have a missing value for that observation.

You can use parentheses without limit to indicate the order in which the elements of the formula should be evaluated. The word GENR is not required; you can omit it and just give the equation you want computed:

```
Z=LOG(REV) ;
```

A complete set of rules and functions (like CNORM(), GAMMA(), etc.) for composing GENR formulas appears in Appendix A. Some common operations are:

+	addition	&	and
-	subtraction	|	or
*	multiplication	~	not
/	division	LOG( )	natural log
**	exponentiation	EXP( )	e to the power
=	equal	SQRT( )	square root
~=	not equal	LOG10( )	base 10 logarithm
>	greater than	ABS( )	absolute value
<	less than	MISS( )	missing value
>=	greater than or equal	<=	less than or equal

Typical examples:

```
SUM = A+B;
AVEQ = Q1*ALPHA + Q2*(1-ALPHA);
```

If you are unsure about the effect of a GENR command and you don't understand the precedence rules in Appendix A, use the PRINT, PLOT, GRAPH, HIST, or MSD commands to check your transformations after performing them.

3.5.1 Dummy variables and recoding

The logical operators such as = ^= > < >= <= & | ^ yield true/false (one/zero) values. This is useful for creating dummy variables and recoding categorical variables. The following example creates a dummy variable that is 1 when X is positive, and zero otherwise:

```
XPOS = X>0;
```

This is much faster than the alternative with SELECT:

```
XPOS = 0; SELECT X>0; XPOS = 1; SELECT 1;
```

See also the DUMMY command (Section 5.6), which is useful for creating large sets of dummy variables and seasonal dummies.

To recode a variable, include logical expressions (0/1) that are multiplied by the value for each case. If the 0/1 expressions are mutually exclusive, only one value will be selected (for the case that is true). For example, say EDUC has 3 categories, and we want to assign them the values 9, 12 and 16 respectively:

```
SCHOOL = (EDUC=1)*9 + (EDUC=2)*12 + (EDUC=3)*16;
```

To recode all negative values to zero, but leave positive values unchanged:

```
X = (X<0)*0 + (X>=0)*X;
```
or
```
X = (X>0)*X;  or  X = POS(X) ;
```

Operations with missing values are a little trickier, because most operations on missing values yield a missing value. For example, to recode missing to -99:

```
SELECT MISS(X); X=-99; SELECT 1;
```

To recode missing values to zero, GENR can be used directly because 0*X is zero, even when X is missing:

```
X = MISS(X)*0 + (.NOT.MISS(X))*X;
```
or
```
X = (.NOT.MISS(X))*X;
```

The operations that can handle missing values without generating a missing answer are the following:

Expression	**Evaluates to**
MISS(@MISS)	True
0*@MISS	0
true .OR. @MISS	True
false .AND. @MISS	False

To recode zeros to missing:

```
SELECT X=0; X=@MISS; SELECT 1;
```

In some computer packages, IF/THEN/ELSE statements are used for these types of transformations, but TSP uses IF/THEN/ELSE for program control. Be careful not to use IF/THEN/ELSE for recoding, since they operate on scalars and not on series.

3.5.2 Lags and leads

A lagged series is indicated by putting the lag in parentheses with a minus sign. For example,

```
FREQ M;
SMPL 72:1,85:12;
DA = AUTOS - AUTOS(-12) ;
```

computes the 12-month change in the monthly series AUTOS. Note that AUTOS must be defined from 1971 through 1985 -- otherwise missing values will be stored in DA. Leads (future values) are defined similarly to lags (past values), except the number in parentheses is positive, e.g. AUTOS(24) is a two-year lead. Lags and leads with series can be used in any command where a plain series may appear. For example,

```
OLSQ Y C,Y(-1);          ? regression of Y on Y lagged once
```

When there is a SMPL gap, lagged values of series still contain the true lagged observations. For example,

```
SMPL 59,70 72,83;
Y=X(-1) ;
```

puts the value of X for the 1971 observation (not the 1970 observation) into the observation of Y for 1972.

Here are some more GENR examples with lags and leads:

```
DP = LOG(P/P(-1)) ;
```

computes the rate of change of P in logs.

```
MOVA = (X(-2) + 2*X(-1) + 3*X + 2*X(1) + X(2))/9 ;
```

computes a moving average of the series X with variable weights, using both lags and leads.

Note for Power Users: Entire equations can also be lagged. See Chapter 9 for information.

3.5.3 Dynamic GENR

If the right-hand side of the GENR equation contains explicit lags (or leads) of the left-hand side series, the series will be updated dynamically. Note that the series must already be defined in the observation(s) preceding the current sample (or following the current sample, in the case of leads). For example,

```
SMPL 1,10; SUM = X; SMPL 2,10; SUM = SUM(-1) + X;
```

creates SUM as the accumulated sum of variable X. This feature can also be used to create autoregressive disturbance terms, dynamic simulations, capital stocks (see also the CAPITL command), and net present values. The dynamic GENR is much faster than using a DO loop to create sums. TSP always prints a note if a dynamic GENR is computed. However, care should be taken to avoid an unintentional dynamic GENR when redefining a series.

For example, the correct commands

```
SMPL 70,85;
MNEW = M(-1); M = MNEW;
or
GENR(STATIC) M = M(-1);
```

replace M with its lag, while the innocent-looking commands

```
SMPL 70,85; M = M(-1);
```

put the 1969 value of M into all observations 1970 through 1985 (probably not what the user intended).

Dynamic GENRs can be computed backwards as well as forwards. For example, a net present value can be computed by using a lead on the right hand side:

```
SMPL 95,2020; NPV = 0;
SMPL 95,2019; NPV = {NPV(+1) + CASH(+1)}/[1+R(+1)];
```

Note that the + sign is not required for leads.

When NPV is computed, a message warns that the dynamic recursion was done in reverse (starting in the year 2019).

3.6 Useful statements at the beginning of a TSP job

The OPTIONS command is handy for customizing various TSP options. Options that you use in most of your programs can be put in a LOGIN.TSP file so they will load automatically (see Chapter 17). For example, if you have a large dataset or model that requires more than the default 4MB of memory, use a command such as:

```
OPTIONS MEMORY=12;
```

at the top of every run.

Plots of actual and fitted values and residuals are available for many statistical procedures in TSP (ACTFIT, OLSQ, 2SLS, LIML, AR1, LSQ, and FIML). Normally they are not printed. The PLOTS option tells TSP to produce plots for subsequent procedures, while NOPLOT stops the plots. Example:

```
OPTIONS PLOTS;
? this is a common customization for regression output
REGOPT(PVPRINT,STARS) T;
OLSQ Y,C,X;
OPTIONS NOPLOT;
```

The HARDCOPY option specifies whether a header is to be printed at the top of each page in the output file. The header contains space for a user name and a 60-character title, which may be varied throughout the run. The default is not to print a header or to page, since it is assumed that these will be added by the program you use to print or display the output, if necessary. Example:

```
OPTIONS HARDCOPY ;
NAME KMARX 'FINAL RESULTS FOR DAS KAPITAL' ;
```

The title can be changed during the job by the TITLE command containing the new title. Example:

```
TITLE 'Results for Transformations' ;
```

TITLE works even without the HARDCOPY option -- the title is just printed directly to the output file, centered and doubly underlined.

More than one option may be specified on an OPTIONS statement. For example:

```
OPTIONS MEMORY=40, PLOTS;
```

The ? character begins a comment -- TSP will ignore everything that follows it until the end of the line. This is useful for reminding yourself what the program is doing at the top and at any critical sections. For example:

```
OPTIONS PLOTS; ? This program tests models with dataset 4.
```

3.7 The order of statements in a TSP job

Although the ordering of a TSP program is very flexible, statements in a job are generally executed in the order in which they appear. Thus, information about inputs must be provided before they can be used. For example, GENR statements for the variables in a particular regression must appear before the OLSQ for that regression. Most users tend to group GENR statements in a job together, but this is not required. A TSP job often has two main parts: a (data analysis) program and a data loading section, separated by an END; statement. In processing the job, TSP starts by reading the program and checking some aspects of it, such as unbalanced parentheses in equations or unterminated DO loops. When it reaches an END; statement it stops reading and begins execution. At this point it has not yet read the data section. When it executes a LOAD; statement in the program, it reads and checks the data until it finds another END; statement or an end of file that marks the end of the data. Then it continues executing the program.

Placing the data in a separate load section is not required. Data can be included with a READ statement anywhere in the program. However, using a data section can be more efficient in terms of memory usage. It is also very useful for suppressing the automatic listing of the data and commands in the output file (with the NOPRINT; command).

3.8 The next step

How you proceed from this point (if you made it this far) depends on your background. If this is your first statistical package, you should read the rest of Chapter 3 and Chapter 4 carefully and then start on your own projects. If you are familiar with statistical packages, you will probably plunge into running TSP without much study, and consult the manual whenever you reach a dead end. The *Reference Manual* is designed to make it easy to look up a command and find out about its options, features, method used and some references for further reading. We recommend that you browse through the manual occasionally; you may be unaware of some of the things TSP can do.

3.9 An extended example

The example introduced here and used throughout the manual is built around a simple illustrative macro model of the U.S. economy. The illustrative model contains equations for five endogenous variables:

CONS:	Consumption in real terms
I:	Investment in real terms
GNP:	Real GNP
R:	Interest rate
P:	Price level--implicit GNP deflator

There are four behavioral equations and one identity:

1. A consumption function where consumption depends only on GNP.
2. An investment function where investment depends on its own lagged value, GNP, and the interest rate.
3. A money demand function relating the interest rate to the velocity of money.
4. A Phillips curve making the price level rate of change depend on its own lagged value, GNP, and a time trend.
5. The GNP identity, stating that GNP equals the sum of consumption, investment, and an exogenous variable called G that measures government expenditures and net exports.

The other exogenous variables of the model are the money supply M, and the time trend TIME.

The first step in the development of the model is to assemble the data. The data section for the illustrative model is typical:

```
FREQ A; SMPL 1946,1975;
READ GNP CONS I ;
475.7 301.4 71
... etc. ...
1186.4 766.6 138.9;
READ EXPORTS ;
11.6 16.6 8.5 8.8 4.0 7.4 4.9 2 4.5 4.7 7.3 8.9 3.5 .9
... etc. ...
END ;
```

Now for some preliminary transformations of the data: first we want to create a trend variable that is one in the first year and increments by one in each succeeding year.

```
TREND TIME ;
```

creates a series called TIME. Next we perform some GENRs:

```
 P = P/100 ; LP = LOG(P) ;
```

The price index with 1972 = 100 is converted to one with 1972 = 1.0 by dividing by 100. The second GENR statement creates a series LP as the natural log of the price level.

```
G = GOVEXP+EXPORTS ;
```

This creates the variable G out of the standard variables GOVEXP and EXPORTS from the National Income Accounts.

```
 R = RS ;
```

At the time the TSP input file was prepared, it was not clear whether a short-term or a long-term interest rate was appropriate. Both were included in the data (RS and RL). In later commands, R is used to stand for the interest rate, whether short-term or long-term. This statement sets R equal to RS. A complete set of results with R equal to RL could be obtained just by changing this statement to GENR R = RL ;

```
PRINT GNP,CONS,I,RS,RL,P ;
```

This statement simply prints these series in a table. It is a good idea to check the transformed data to see that the manipulations were correctly specified.

```
SMPL 1948,1975;
MNEW = (M+M(-1))/2 ; M = MNEW;
```

Now we change the sample to begin in 1948 rather than 1946 because we are going to use some lagged data in the computations. When we define the lagged price change (DP1) below, we will need observations lagged twice, so we start in 1948 rather than 1947. In the published data, M is the money stock at the end of the year. Money in this model is the average during the year. This is approximated by averaging the year-end value and the value at the end of the previous year.

```
DP = LOG(P/P(-1)) ;
DP1 = DP(-1) ;
```

DP1 is the lagged rate of inflation.

```
SMPL 1949,1975 ; DM = LOG(M/M(-1)) ;
```

DM is the rate of change of the money stock. Multiplying DM by 100 would give the percent rate of change. Because the first observation of the money stock was zero, we need to start in 1949 to obtain good values of DM -- remember that M was redefined as a two year moving average earlier.

```
LGNPCUR = LOG(GNP*P) ;
```

LGNPCUR is the log of GNP in current prices.

With the data prepared, we can begin estimation of the equations of the illustrative model by ordinary least squares and two-stage least squares. For the consumption equation, the TSP command for regression is the following:

```
OLSQ CONS C,GNP ;
```

OLSQ computes the simple regression of CONS on GNP with an intercept. Note how the intercept is specified: C is a built-in series in TSP consisting entirely of 1's. C is used mainly to specify a constant or intercept in regressions, but may be used anywhere that a series is valid in any TSP statement. If C does not appear in the list of variables for a regression, the regression will be estimated without an intercept; that is, the regression line will be forced through the origin. In this respect, TSP is different from some other regression programs.

The commands

```
FRML MPC GNP -1;
ANALYZ MPC;
```

define a hypothesis to be tested (that the coefficient of GNP is equal to one) and then test it with a t-test performed by ANALYZ

To compute two-stage least squares estimates for the simple consumption function, use:

```
LIST IVS C,G,LM,TIME;
2SLS(INST=IVS) CONS C,GNP;
```

The LIST command defines a TSP variable list (in this case the list of instruments for the model, which will be used repeatedly in 2SLS and LSQ). The INST= option in 2SLS specifies the list of instrumental variables for this equation -- note that the constant, C, must be specified as an instrument explicitly. Any exogenous variable that appears in the equation must also be in the list of instruments; in this case, C is the only such exogenous variable.

We also estimate this equation with LIML (Limited Information Maximum Likelihood), which is specified in the same way as 2SLS:

```
LIML(INST=IVS) CONS C,GNP;
```

Estimation of the investment equation is next. We hypothesize that investment depends on the demand for capital services, which is the ratio of real GNP to the rental price of capital. The rental price of capital, in turn, is the sum of the rate of depreciation and the interest rate. The following statements form the appropriate variables and carry out the estimation.

```
CONST DELTA 15 ;
```

defines a constant, DELTA, interpreted as the rate of depreciation and assigned the value 15 (% per year).

```
GENR GNPN = GNP/(DELTA+R) ;
```

creates the capital demand variable, GNPN. Note that constants may appear in GENR statements (see also Section 6.1).

```
OLSQ I I(-1),GNPN ;
```

estimates the investment equation. It is followed by a nonlinear least squares estimation of the same equation as an illustrative example. However, since the equation in this version is really linear, there will be no difference in the estimates. If DELTA has been specified as a parameter rather than a constant, the estimation would be truly nonlinear.

The next OLSQ and 2SLS commands estimate a constrained version of the interest-rate equation where the coefficients of LGNPCUR and LM are required to be the same in magnitude but opposite in sign. The constraint is imposed by combining them in LVELOC, the log of monetary velocity.

```
LVELOC = LGNPCUR-LM ;
OLSQ R C,LVELOC ;
2SLS(INST=IVS) R C,LVELOC ;
```

The final OLSQ and 2SLS commands estimate the inflation equation.

```
OLSQ DP DP1,LGNP,TIME,C ;
2SLS(INST=IVS) DP DP1,LGNP,TIME,C ;
```

OLSQ and 2SLS statements can appear in any order, not just in pairs as in this example. It is usually more convenient to group the estimation statements for a particular equation so that the output from the estimations appears adjacently.

Here is the TSP job for the illustrative model, which includes several commands like LSQ, FIML, ANALYZ, and SIML that will be introduced in later chapters. We have not included the full data set, but you can find the complete program and data on the web site.

```
OPTIONS CRT;
NAME ILLUS50
'Sample Run for TSP 5.0: Illustrative Model from Manual' ;
?
FREQ A;          ? Annual frequency.
SMPL 46 75 ;   ? Sample is 1946 to 1975.
LOAD ;           ? Read in data.
TREND TIME ;
?
?  Data Transformations.
?
P = P/100 ; LP = LOG(P) ;
G = GOVEXP+EXPORTS ;
R = RS ;
PRINT GNP CONS I RS RL P ;
SMPL 48,75 ;
MNEW = (M+M(-1))/2 ; M = MNEW;
DP = LOG(P/P(-1)) ; DP1 = DP(-1) ;
SMPL 49,75 ; DM = LOG(M/M(-1)) ;
PRINT G M DM TIME DP ;
LGNPCUR = LOG(GNP*P) ; LM = LOG(M) ;
LVELOC = LGNPCUR-LM ; LGNP = LOG(GNP) ;
?
? Data is now generated; Try estimation on single equations.
?
SMPL 49,75 ;
OPTIONS PLOTS ;
TITLE 'OLSQ - Ordinary Least Squares' ; PAGE ;
OLSQ CONS C,GNP ;
?
? Test the hypothesis that the marginal propensity to consume
? is one.
FRML MPC GNP-1 ;
ANALYZ MPC ;
?
? Form a consumption equation for model estimation later.
FORM(COEFPREF=B) CONSEQ;
TITLE 'PRINT Example from Users Guide' ; PAGE ;
PRINT B1 CONSEQ @VCOV @RES;
?
```

```
TITLE 'Engle-Granger Test for Cointegration' ; PAGE ;
SMPL 50,75 ;
DU = @RES-@RES(-1) ;
? Regress diff. residuals on trend and lag residual.
OLSQ(SILENT) DU C TIME @RES(-1) ; SET TSTAT = @T(3);
? Test with Dickey-Fuller distribution
CDF(DICKEYF,TREND,DF=@NOB,PRINT) TSTAT ;   ?
?  Plot the residuals from the consumption equation.
?
SMPL 49,75 ;
GENR CONSEQ CONSFIT ; RES = CONS-CONSFIT ;
PLOT (MIN=-25,MAX=25,VALUES,HEADER,BAND=STANDARD,
        INTEGER,BMEAN)RES * ;
?
?  Estimation with correction for first order serial correlation.
?
AR1 CONS C,GNP ;
FORCST(PRINT) CONSP ;
OPTIONS NOPLOT ;
?
?  Two stage least squares.
?
LIST IVS C G LM TIME ;        ? Define a list of instrumental variables
?
TITLE 'INST - Instrumental Variable Estimation' ; PAGE ;
2SLS(INST=IVS) CONS C,GNP;
?
TITLE 'LIML - Limited Information Maximum Likelihood Estimation'
PAGE ;
LIML(INST=IVS) CONS C,GNP;
PAGE;
?
?  Linear Estimation of Investment Equation.
CONST DELTA 15 ; GNPN = GNP/(DELTA+R) ;
OLSQ I I(-1),GNPN ;
FORCST(PRINT,DYNAM,DEPVAR = I) IFIT ;
?
?  'Nonlinear' Estimation of the same equation by OLS and 2SLS.
FRML INVEQ I = LAMBDA*I(-1) + ALPHA*GNP/(DELTA+R) ;
PARAM LAMBDA .05 ALPHA .2 ;
```

```
LSQ(PRINT) INVEQ ;
LSQ(INST=IVS) INVEQ ;
?
?  Linear Estimation of Interest Rate Equation.
OLSQ R C,LVELOC ;
2SLS(INST=IVS) R C,LVELOC ;
PARAM D F;
UNMAKE @COEF D F;           ? Starting values for INTRSTEQ below.
?
?  Linear Estimation of Price Change Equation.
OLSQ DP DP1 LGNP TIME C ;
2SLS(INST=IVS) DP DP1,LGNP,TIME,C ;
PARAM PSI PHI TREND P0 ;   ? Parameters of price change eqn.
? Set starting values for PRICEQ below.
UNMAKE @COEF PSI PHI TREND P0 ;
?
?  These are the equations of the simultaneous equations model.
?
IDENT GNPID GNP-CONS-I-G ;
?  The next equation was made by FORM earlier in the run.
? FRML CONSEQ CONS = B0 + B1*GNP ;
FRML INTRSTEQ R = D + F*(LOG(GNP)+LP-LM) ;
FRML PRICEQ LP = LP(-1) + PSI*(LP(-1)-LP(-2)) + PHI*LOG(GNP)
          + TREND*TIME +P0 ;
?
? Estimate by multivariate regression, assuming contemporaneous
? correlation of the residuals.
?
LSQ(STEP = CEAB) CONSEQ INVEQ INTRSTEQ PRICEQ ;
PRINT @FIT ;
?
?  Estimate by nonlinear three stage least squares.
?
TITLE '3SLS - Three Stage Least Squares' ; PAGE ;
3SLS(PRINT,INST=IVS) CONSEQ INVEQ INTRSTEQ PRICEQ ;
?
?  Estimate by Full Information Maximum Likelihood.
?
FIML(ENDOG=(GNP,CONS,I,R,LP)) GNPID CONSEQ INVEQ
          INTRSTEQ PRICEQ ;
```

```
?
?  Test some nonlinear hypotheses about Long run coefficients.
?
FRML LR1 ALPHALR = ALPHA/(1.-LAMBDA) ;
FRML LR2 PHILR = PHI/(1.-PSI) ;
PAGE;
ANALYZ LR1 LR2 ;
PAGE;
?
? Simulate the model over the second half of the sample
? and plot the results.
?
SMPL 66 75 ;
SIML (PRNSIM,PRNDAT,TAG = S,ENDOG = (GNP,CONS,I,R,LP))
   CONSEQ INVEQ INTRSTEQ GNPID PRICEQ ;
PLOT (RESTORE, MAX=1500, MIN=500, LINES=(1000)) GNP G
          GNPS H CONS C CONSS D ;
PLOTS ;
ACTFIT R RS ;
END ;
?
? START OF THE DATA SECTION
?
NOPRINT;
SMPL 46 75 ;
LOAD GNP CONS I ;                  ? GNP, CONSUMPTION,
          INVESTMENT
475.7 301.4 71
468.3 306.2 70.1
487.7 312.8 82.3
... data continues...
1186.4 766.6 138.9
;
LOAD EXPORTS ;                     ? EXPORTS
11.6 16.6 8.5 8.8 4.0 7.4 4.9 2 4.5 4.7 7.3 8.9 3.5 .9
5.5 6.7 5.8 7.3 10.9 8.2 4.3 3.5 -.4 -1.3 1.4 -.6 -3.3
7.2 16.6 23.5 ;

LOAD GOVEXP ;                      ? GOVERNMENT
          EXPENDITURES
```

```
91.8 75.4 84.1 96.2 97.7 132.7 159.5 170.0 154.9 150.9
152.4 160.1 169.3 170.7 172.9 182.8 193.1 197.6 202.7 209.6
229.3 248.3 259.2 256.7 250.2 249.4 253.1 252.5 254.3 257.4
 ;

...reads in M, P, RS -- full data set available on the web site...

LOAD RL ;                          ? LONG TERM INTEREST RATE.
2.53 2.61 2.82 2.66 2.62 2.86 2.96 3.2 2.9
3.06 3.36 3.89 3.79 4.38 4.41 4.35 4.33 4.26 4.4
4.49 5.13 5.51 6.18 7.03 8.04 7.39 7.21 7.44 8.57 8.83
 ;
END ;
```

4. INTERACTING WITH TSP

This chapter provides a general overview of TSP's interactive mode. It is a sufficient introduction for new users anxious to dive into their first session. You will find additional convenience features discussed in Chapter 17, and further information about the commands in the *Reference Manual.* A short sample session has been included at the end of the chapter for added clarity (**Figure 4.1**). If you use Windows, you may want to try the visual user interface utility *Givewin*,™ which allows both batch and interactive modes of running TSP.

4.1 *Basic operation*

You don't really need to know much before jumping into your first interactive session, but it's useful to have an idea of how to start, stop, and enter commands. Also, having an understanding of the operating modes of the program may affect how you approach it and will enable you to get the most out of it.

4.1.1 Beginning and ending a session

The command you use to start running interactive TSP depends on how the program was installed on your system; it will probably be "TSP" (on some systems, you double-click on a TSP icon). If this doesn't work you will have to consult your system manager. Once you have given the command (or double-click), the following will appear:

Enter batch filename [or press Enter for interactive]:

Two responses are possible:

If you supply a filename here (such as ***prog2a***), TSP will run in *batch* mode (in this case, reading commands from ***prog2a.tsp*** and putting output in the file ***prog2a.out***). When TSP finishes, it will prompt for a new batch filename (or to rerun the same batch file). If you are using a multitasking operating system (such as unix, OS/2, Windows, or Mac), you can use a task switch key at this point to view the output file and edit the input file, and then rerun TSP a second time if necessary. Otherwise, you'll have to exit from TSP, give commands to do these operations, and then restart TSP. (In this case, you would be better off using the command *tsp prog2a* which will run TSP in batch mode automatically and skip the prompt). Batch mode is recommended if a long sequence of TSP commands is needed to estimate your model; it is easier to revise and reproduce results this way.

If you press Enter (or Return on some keyboards) to the initial prompt, the following will appear:

```
Enter TSP statements:
1?
```

You are now in the *interactive* mode of the program, and each command that you enter will be executed immediately, and results will be displayed on the screen.

It is possible to suppress execution of commands selectively; see the next section and COLLECT in the *Reference Manual*. It is also possible to redirect output; see Section 4.4 or the OUTPUT command in the *Reference Manual*.

If your last session ended abnormally, the following message should appear before the first line number prompt.[7]

```
WARNING> Your previous TSP session was terminated abnormally.
Do you wish to recover it (y/n)? [y]
```

To end an interactive TSP session use either the EXIT or STOP command. But first be sure to save any output you want. SAVE (see Section 4.3) saves all the current TSP variables (but not the commands) so they can be restored later. The batch-compatible commands that you entered will be saved in the file ***BKUP.tsp***. This file may be useful for starting your next session (with INPUT or ***login.tsp***). You can restart an interactive session (i.e. start over at line 1), by using the CLEAR command. You can also use the SYSTEM command to temporarily suspend your session to take care of other business on the computer. See Chapter 17 for more information.

4.2 Modes of operation

TSP operates in several modes, all of which can be accessed by the Givewin shell (see below in this section):

Interactive: One line at a time is executed, right after the command is entered.

[7] Chapter 17 discusses the automatic recovery system. See also the RECOVER section in the *Reference Manual*.

Control may be transferred between interactive and collect modes as often as you like. In general, blocks of statements that are executed as a unit (PROCs, DO loops, etc...) must be entered in collect mode. EXIT or STOP returns you to the operating system, terminating TSP. CLEAR restarts an interactive session (without saving anything). It is also possible to pass control temporarily to the operating system without interrupting the interactive session (SYSTEM command.)

Collect: Collect mode is entered from interactive mode by the COLLECT command. Execution is suppressed until a group of commands has been entered, and execution is requested (EXEC). The commands are then processed and you are returned to interactive mode. EXIT returns you to interactive mode without executing the collected lines. Reading a stream of commands from an external file (with INPUT) is functionally the same as COLLECT. With INPUT, execution is automatic when END or end-of-file is reached. In either case, the commands read are incorporated into the interactive session, and may be REVIEWed, EDITed, etc... later in the session.

Batch: In batch mode TSP reads and executes a previously prepared, complete TSP input file. You create this input plain text (ASCII) file with a separate text editor (such as the EDIT command in DOS, or a word processing program like Microsoft Word, saving as a text only file). You can run your file via a batch queue (multi-user systems), or while waiting at your PC/terminal. In either case, TSP usually terminates when execution of the file is completed.

You may want to execute a short TSP program you have prepared, displaying the output on the monitor. In this case, you could open TSP interactively, and then immediately INPUT a previously prepared batch file.

Givewin (for Windows only): This visual interface shell works just like any other Windows program. It allows you to edit multiple TSP input files using standard Windows editing conventions (such as cut and paste). You can run the files in batch mode by clicking a run button, and browse and print the output. You can also use TSP interactively within Givewin; either way, operation in Givewin means that you have access to its graphics capabilities for plotting from within TSP.

4.2.1 Entering commands in Interactive Mode

There is not much difference between entering TSP commands in a file for batch execution, and entering them in interactive mode -- just follow the usual syntax. The main difference is that semicolons at the end of a line are not necessary -- a carriage return defines the end of the command. You may indicate that a statement takes more than one line by typing a backslash (\) at the end of the line(s) that are to be continued.

However, statement-ending semicolons are still required in batch or INPUT files that you use in the interactive session. Also, if you REVIEW any portion of an interactive session, you will notice that they have been added for you. Of course, you may enter more than one command per line in interactive mode if you separate them with semicolons.

There are some features of TSP that may be especially useful when working interactively. The first is that you can abbreviate TSP commands (and options). The abbreviation must be unambiguous, and must not skip characters; it may be any length (including 1). For example:

OLS	valid abbreviation for OLSQ
BJ	ambiguous -- could be BJEST, BJIDENT, BJFRCST
BJF	valid for BJFRCST
Q	valid for QUIT (no other "Q" commands)
FCST	invalid for FORCST (skips characters)

You can use cursor keys to recall and edit lines that you've previously typed. This is useful for correcting typographical errors, adding/dropping variables, or repeatedly executing commands. The cursor keys work just like they do in DOS EDIT, Notepad, or Wordpad:

Key	**Action**
↑	recall previous line(s)
↓	recall later line(s)
→	move (cursor) right in current line
←	move (cursor) left in current line
Home	move (cursor) to start of line
End	move (cursor) to end of line
Ins	toggle typeover/insert mode (default: typeover)
Del	delete character at cursor
Esc	clear current line
Backspace	delete character to left of cursor

For other methods of editing commands in interactive mode, please see Chapter 17.

4.2.2 Requesting information: HELP, SHOW, etc.

Several commands will give information on the current status of your session or remind you of command syntax.

HELP *TSP command*

displays information about the syntax and function of a particular TSP command. HELP by itself gives a list of TSP commands. If you are using a Windows version of TSP, you can also obtain help via the TSP Help System that is shipped with the program.

REVIEW *first line, last line*

redisplays a range of lines from your session.

FIND *TSP command*

displays all the lines in your session that begin with the specified TSP command. This is useful when you want to EDIT or EXEC a command, but you don't know its line number.

SHOW *list of names and/or keywords*

provides information about how TSP has stored specific items, or groups of items in the symbol table. You can SHOW SERIES to find out which time series you have stored, along with their frequencies, starting and ending dates, and number of observations. You can also SHOW SMPL or FREQ to find out the current settings of each.

DIR *

displays all the files in your current directory with the extension TSP; this can be useful in conjunction with the INPUT command. DIR "alone" prompts for a directory specification; you can display information about any file, from any directory (as well as request information regarding size, date, etc.).

4.2.3 Methods of entering or reading data

The interactive mode of TSP offers a number of approaches to loading data. If the data you need is not on disk in some usable form, the easiest method of data entry is the ENTER command.

ENTER *seriesname*

prompts you to enter data based on the current sample and frequency. Any number of data points may be entered per line. If a non-numeric item is encountered (your fingers slip on the keyboard) it is ignored (along with any numbers that follow it on the line). Numeric corrections to the values of a series may be made with the UPDATE command. ENTER prompts for data until enough numbers have been supplied to fill the current SMPL.

READ *seriesname*

also allows you to enter data from the keyboard, though no prompting will occur (other than the current line number prompt), and nothing is stored until a semicolon is entered; this is one of the few instances in which a semicolon is required in interactive mode. Numeric errors not corrected immediately can also be UPDATEd.

If the data you need is already stored on disk, you can use any of the following methods to read it:

READ *with the options FILE= and/or FORMAT=*
INPUT *the name of a file that contains a data section*
IN *databank*
FETCH *micro-TSP (or EViews) databank*
RESTORE *a SAVEd session*

4.2.4 Saving selected output: OUTPUT, TERM, PrtSc

As you use TSP interactively, you will often produce output on the screen that you want to save for later examination or printing. You can save screen output easily with the OUTPUT command.

OUTPUT *filename*

sends all output to the file "filename.out" until the TERMINAL command, or another OUTPUT command is encountered. You can create as many output files as you like during a session; each OUTPUT statement opens a new output file and closes the previous one (if there was one). If the output filename already exists (whether it was created earlier in the session or not), output will be appended to the old file rather than creating a new file. For rules on specifying filenames, see OUTPUT in the command reference section.

It is not possible to send output to the terminal and an output file simultaneously, but error and warning messages are displayed in both places so you will know if something is seriously wrong. There are several solutions to the "seeing your output and saving it, too" dilemma. Which is best depends on how much of the output you want to see, and how much you are paying for computer resources. For simplicity, the following examples save output from one procedure only; all may be extended to handle any volume of output.

To print just the output on the screen, you can use the shift-printscreen keys in either DOS or in a DOS window running TSP. You can also mark the screen output in a DOS window and save it in Clipboard for later use.

If you are not certain that you want to save the results, you can execute the procedure(s) first, and then repeat the command after issuing an OUTPUT command if it looks good:

```
10? OLSQ CONS C GNP
 (regression output)
11? OUTPUT REGRESS  (changes output to a file names REGRESS)
12? EXEC 10            (re-executes line 10, the OLSQ)
13? TERM               (switches back to terminal output)
```

3. If you only need to see selected results from what you save, the following method is faster. All output is sent to the file until TERM, and any other variables created during execution may be examined with PRINT or SHOW.

```
10? OUTPUT REGRESS
11? OLSQ CONS C GNP
12? TERM
13? TSTATS(NAMES=@RNMS) @COEF @VCOV  (Prints t-statistics)
```

4. This method takes the most steps, but avoids a wait during re-execution. The system prompt character is shown as $. This may be different, for example, on a PC the prompt is the disk drive letter (usually C> or D>):

```
 10? OUTPUT REGRESS
 11? OLSQ CONS C GNP
 12? TERM
 13? SYSTEM
```

```
$ TYPE REGRESS.OUT
(contents of regress.out)
$ CONTINUE
14?
```

You can look at the file with your editor instead of the TYPE command. You can also print, delete, rename, or copy the file before continuing the use of TSP. The TERM command is essential because it closes the file; if the file is not closed when you try to look at it, it will either appear empty or be missing the output you added since the last time it was closed.

4.2.5 Sample session in interactive mode

The following pages give a simple example of running TSP in interactive mode. For information on advanced techniques in interactive mode, please see Chapter 17.

```
Enter TSP statements:

  1 ? input illusdat

Do you want the output displayed at the terminal (y/n)?
[y]

Current sample:  1961 to 1975
  6 ? review

       1. ? input 'illusdat';
       1. FREQ A;
       2. SMPL 61 75 ;
       3. LOAD GNP CONS I ;
       4. ? GNP, CONSUMPTION, INVESTMENT
       4. REGOPT(LMLAGS=2) LMAR ;
       5. ? TURN ON LMAR DIAGNOSTIC
       6. REVIEW;

  7 ? olsq cons c gnp

                    Equation   1
                    ============

      Method of estimation = Ordinary Least Squares


Dependent variable: CONS
Current sample:  1961 to 1975
Number of observations:  15

         Mean of dep. var. = 626.527
    Std. dev. of dep. var. = 105.195
  Sum of squared residuals = 1911.53
     Variance of residuals = 147.040
  Std. error of regression = 12.1260
                 R-squared = .987662
        Adjusted R-squared = .986712
              LM het. test = .513322 [.474]
             Durbin-Watson = .616923 [.000,.002]
Breusch/Godfrey LM: AR/MA1 = 9.22897 [.002]
Breusch/Godfrey LM: AR/MA2 = 8.69666 [.013]
          Jarque-Bera test = .659462 [.719]
           Ramsey's RESET2 = 6.96993 [.022]
           F (zero slopes) = 1040.62 [.000]
            Schwarz B.I.C. = 60.3492
            Log likelihood = -57.6411

           Estimated    Standard
Variable  Coefficient     Error       t-statistic   P-
```

```
value
C         -63.3408      21.6135       -2.93061
[.012]
GNP       .676823       .020981       32.2586
[.000]
  8 ? plot @res
```

This plot is produced in a separate window and is shown at the end of this run.

```
  9 ? normal gnp 72 2608.5

 10 ? normal cons 72 1621.9

 11 ? update gnp

Which observations do you wish to update? 74 85

Enter data for GNP
      1974? 2729.3 2695.0 2826.7

      1977? 2958.6 3115.2 3192.4

      1980? 3187.1 3248.8 3166 3277.7 3492 3573.5

 12 observations of GNP      have been updated.

 12 ? update cons

Which observations do you wish to update? 74 85

Enter data for CONS
      1974? 1674 1711.9 1803.9

      1977? 1883.3 1961 2004.4 2000.4

      1981? 2024.2 2050.7 2145.9 2239.9 2312.6

 12 observations of CONS     have been updated.

 13 ? save

 14 ? graph gnp cons    ? Plot Consumption vs. GNP
```

This graph is produced in a separate window and is shown at the end of this run.

```
 15 ? find olsq   ? Look for all OLSQ commands thus far
       6. OLSQ CONS C GNP;
 16 ? exec 6      ? Repeat the OLSQ under the new SMPL
       6. OLSQ CONS C GNP;

      Equation   2
      ============
      Method of estimation = Ordinary Least Squares
 Dependent variable: CONS
 Current sample:  1961 to 1985
 Number of observations:  25

          Mean of dep. var. = 1649.17
     Std. dev. of dep. var. = 385.115
   Sum of squared residuals = 19485.3
      Variance of residuals = 847.185
   Std. error of regression = 29.1064
                  R squared = .994526
         Adjusted R squared = .994288
               LM het. test = .634874 [.426]
              Durbin Watson = .993869 [.001,.006]
 Breusch/Godfrey LM: AR/MA1 = 5.96885 [.015]
 Breusch/Godfrey LM: AR/MA2 = 6.52117 [.038]
           Jarque Bera test = 1.24251 [.537]
            Ramsey's RESET2 = 3.64755 [.069]
            F (zero slopes) = 4178.60 [.000]
             Schwarz B.I.C. = 6.91605
             Log likelihood =  118.705

            Estimated    Standard
 Variable  Coefficient     Error       t statistic   P
value
 C          190.235      29.0447        6.54975
[.000]
 GNP       .694874       .010750       64.6421
[.000]

17 ? retry 6         ? Change from OLSQ to AR1
       6. OLSQ CONS C GNP;
      >> r olsq ar1
       6. AR1 CONS C GNP;

                   Equation   3
                   ============
      FIRST ORDER SERIAL CORRELATION OF THE ERROR
         MAXIMUM LIKELIHOOD ITERATIVE TECHNIQUE

     CONVERGENCE ACHIEVED AFTER   11 ITERATIONS
 Dependent variable: CONS
 Current sample:  1961 to 1985
 Number of observations:  25

(Statistics based on transformed data) (Statistics based on original
data)
       Mean of dep. var. = 807.195        Mean of dep. var. = 1649.17
  Std. dev. of dep. var. = 171.116   Std. dev. of dep. var. = 385.115
Sum of squared residuals = 14785.7 Sum of squared residuals = 15034.9
   Variance of residuals = 642.855    Variance of residuals = 653.692
Std. error of regression = 25.3546 Std. error of regression = 25.5674
               R squared = .979573                R squared = .995846
      Adjusted R squared = .978685       Adjusted R squared = .995666
           Durbin Watson = 1.67542            Durbin Watson = 1.66861
Rho (autocorrelation coef.) = .536778
      Standard error of rho = .176223
        t statistic for rho = 3.04601
             Log likelihood =  115.425

            Estimated    Standard
 Variable  Coefficient     Error       t statistic   P value
 C          155.104      46.8728        3.30903      [.001]
 GNP       .682614       .017270       39.5257       [.000]
```

```
 18 ? exit     ? Terminate the session
```

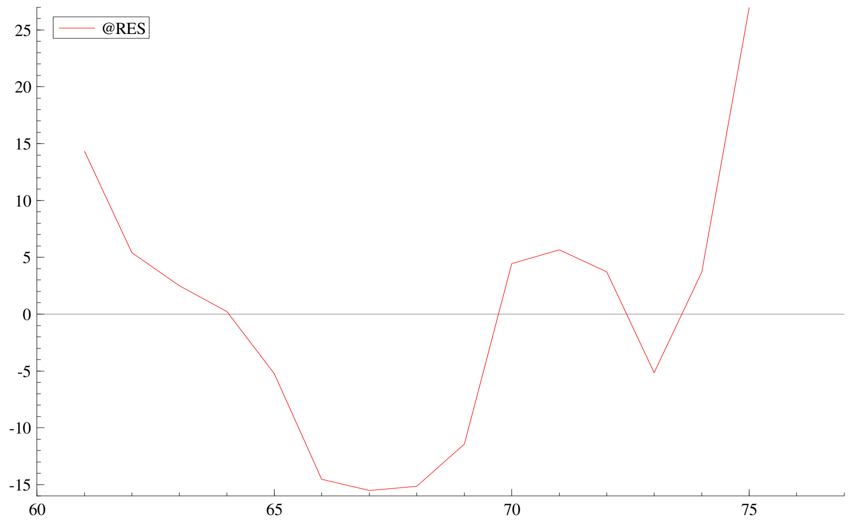


Figure 4.1 Sample interactive session PLOT output

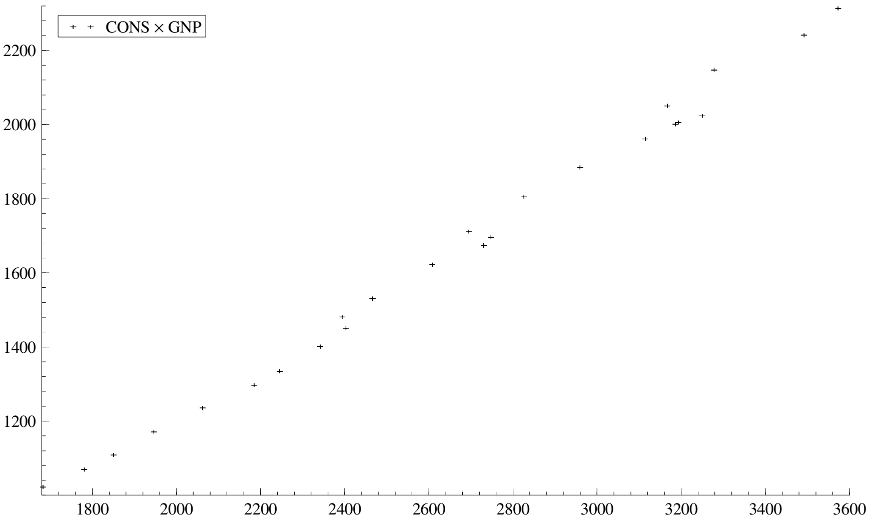


Figure 4.2 Sample interactive session GRAPH output

5. ESTIMATION OF LINEAR EQUATIONS

This chapter discusses several simple statistical models that involve linear structural equations. The models include: descriptive univariate and bivariate statistics (MSD), ordinary least squares (OLSQ), two-stage least squares (2SLS), limited information maximum likelihood (LIML), least squares with correction for serial correlation (AR1), and quantile estimation (LAD).

In these procedures the equation is defined implicitly by listing its variables. The observations used for estimation are defined by the current sample (the most recent SMPL, SELECT, or SMPLIF). Any observations containing missing values are temporarily dropped from the sample for all procedures except AR1 with the ML option. Most of the models allow the WEIGHT option for general least squares estimates, and the ROBUST option for standard error computations robust to heteroskedasticity. They also allow distributed lag variables (polynomial, Shiller, or unconstrained). You can use REGOPT (described fully in the *Reference Manual*) to control the calculation and printing of numerous diagnostic statistics.

Chapter 9 discusses estimation with qualitative dependent variables (PROBIT, TOBIT, LOGIT, SAMPSEL). Estimation methods for nonlinear and simultaneous equation (linear and nonlinear) models are covered in Chapter 7 (LSQ, FIML, and GMM) and in Chapter 9 (ML). Chapter 8 describes constructing hypotheses tests for linear and nonlinear models. Chapter 11 discusses estimation of time series models (ARIMA models, VAR and GARCH-M estimation).

5.1 Descriptive Statistics: MSD, CORR

Before doing any regressions or using the more expensive (in terms of computer time) estimation procedures, check your input and transformed data: you can look for bad observations or outliers introduced by data entry error and check that the data transformations did what you expected. If you don't have much data, you can check it by using the PRINT command, but this method may be overwhelming if you have several hundred observations.

TSP has several descriptive statistics procedures that conveniently summarize large amounts of data for you. MSD prints a table of univariate statistics. MSD (CORR) computes and prints a correlation matrix. MSD(COVA) produces a covariance matrix. MOMENT computes an uncentered sum of squares and cross products matrix. See Chapter 6 for a description of the graphics procedures, including HIST, which prints and plots frequency bar charts.

MSD followed by a list of series will produce a table of means, standard deviations, sums, variances, skewness, kurtosis, and maximum and minimum values for all the series. Example:

```
MSD LSALES,LGNP;
```

The CORR, COVA, and MOMENT options work the same way, except that they produce a matrix for the list of series provided. The moment of two variables is their uncentered covariance -- the sum of cross products divided by the number of observations.

```
MSD(CORR) LSALES,LGNP ;
```

would store the 2x1 vectors @MEAN, @STDDEV, @MIN, @MAX, @SUM, @VAR, and the 2x2 matrix @CORR. (See section 6.1 for more information about using @ variables.) The results computed by these commands are stored for use in further analysis.

You can use the WEIGHT= option in MSD to provide a weighting variable for the computations. (See Section 5.8.)

To obtain the median and interquartile range of the series use the ALL option of the MSD command:

```
MSD (ALL) LSALES LGNP ;
```

5.2 Ordinary least squares: OLSQ

The first variable specified in the OLSQ command is the dependent variable and the rest are the independent variables. Recall, for example, the consumption function in the illustrative model:

$$CONS=a+b*GNP$$

a is a parameter that multiplies an implied variable, the constant, which always has the value 1. In TSP, the constant has the special name, C (or CONSTANT). It may be used in OLSQ or elsewhere without loading it as data. The OLSQ statement for our consumption function is

```
OLSQ CONS C,GNP ;
```

OLSQ computes the least squares regression coefficients and a variety of associated statistics. These statistics include the standard error of the residuals from the regression, the Durbin-Watson statistic, and the mean and standard deviation of the dependent variable. The R-squared, R-squared adjusted for degrees of freedom, and F-statistic for the hypothesis that all the coefficients except the constant are zero are also printed.

You can plot the actual and fitted values of the dependent variable by including PLOTS earlier in the program.

5.3 Regression output

All regression procedures in TSP produce output in the same format. We describe the output here using the OLSQ command as an example (**Figure 5.1**). In later sections of the manual we will point out any differences you may expect when using other estimation methods.

Using the standard textbook notation where y_t is the dependent variable and X_t is the vector of independent variables, e_t (the residual) is defined as

$$e_t = y_t - \hat{y}_t \qquad \text{where } \hat{y}_t = X_t b$$

and where b denotes the estimated regression coefficients. To give the exact formulas for the regression statistics in the TSP output, we define the sum of squared residuals **SSR** and the total sum of squares **SST**:

$$SSR = \sum_{t=1}^{T} e_t^2 = \sum_{t=1}^{T} (y_t - \hat{y}_t)^2$$

$$SST = \sum_{t=1}^{T} (y_t - \overline{y})^2$$

The standard error of the regression is then given by

$$s = \sqrt{SSR/(T-K)}$$

where T is the number of observations in the current sample and K is the number of independent variables.

The **R-squared** is defined as the squared correlation coefficient between y and *yhat*:

$$R^2 = \frac{[\mathrm{cov}(y,\hat{y})]^2}{\mathrm{var}(y)\cdot \mathrm{var}(\hat{y})}$$

R-squared is defined the same way for the 2SLS, AR1, LIML and PROBIT commands; only the formulas for *yhat* and *b* are different. For an OLSQ regression with a constant term, this squared correlation is equal to 1-**SSR/SST**. If *yhat* is constant, the R-squared will be zero. On the other hand, if the regression yields a perfect fit of the dependent variable, the correlation and the R-squared will be one; thus you can interpret the R-squared as the fraction of total variance that is "explained" by the variables in your regression other than the constant. It is also the squared cosine of the angle between the actual and predicted *y* and *yhat*, once the means of *y* and *X* have been removed.

The **adjusted R-squared** is the quantity called R-bar-squared in some texts. The formula for it is given by:

$$\overline{R}^2 = \frac{(T-1)R^2 + 1 - K}{T-K}$$

For a regression with a constant term this can also be written as

$$\overline{R}^2 = 1 - \frac{SSR/(T-K)}{SST/(T-1)}$$

The **adjusted R-squared** has the advantage that it does not automatically increase as variables are added to the regression since the numerator includes an adjustment for the number of estimated coefficients (*K*). As more variables are added to the regression with little or no additional explanatory power, it is possible for this quantity to become negative.

The next few statistics in the output report the results of tests for the validity of several assumptions of the linear regression model: homoskedasticity, lack of serial correlation, correct functional form, and normality.

The **LM heteroskedasticity test** is a test for homoskedasticity of the residual variances. It is computed by regressing the squared residuals on the squared fitted values of the regression. The resulting TR^2 has a chi-squared distribution with one degree of freedom.

The **Durbin-Watson statistic** is described in Durbin and Watson (1951). A useful discussion of its properties and how to interpret it is in Pindyck and Rubinfeld (1991). The formula used by TSP to compute this statistic is

$$DW = \frac{\sum_{t=2}^{T} (e_t - e_{t-1})^2}{SSR}$$

This statistic is valid even if there are gaps in the SMPL (missing observations), although in that case, it will be somewhat conservative. The p-values printed next to the Durbin-Watson statistic are exact finite sample values when the data are time series, and are an asymptotic approximation when the data have frequency *N*. See REGOPT in the *Reference Manual* for details.

If a lagged dependent variable is included in the right-hand-side variables, the Durbin-Watson statistic is biased, so TSP prints two alternate statistics described in Durbin (1970). These two statistics are labeled "Durbin's h" and "Durbin's h (alt.)" respectively and they are asymptotically equivalent. Sometimes ***h*** cannot be computed due to a negative square root. On the other hand, the ***h*** alternative may have little power in small samples.

Ramsey's RESET test is a Lagrange multiplier test of functional form, computed by regressing the residuals on the independent variables and the square of the fitted dependent variable. This version of the test has a chi-squared distribution with one degree of freedom. The RESET test can also be significant if there is a single outlier that is fit well by a quadratic function of the right hand side variables *X*.

The **Jarque-Bera test** is a joint test for skewness and kurtosis of the disturbances, asymptotically distributed as chi-squared random variable with two degrees of freedom under the null of normality.

The individual t-statistics produced by TSP correspond to variable by variable tests of exclusion restrictions: they are computed as the value of the coefficient divided by its estimated standard error. This quantity is distributed as a student's t variable with (*T-K*) degrees of freedom. The t-statistic produced by TSP represents a test of only one of many possible hypotheses about the coefficient estimate: that it is zero. You may wish to test a different hypothesis, i.e., that the coefficient is equal to a_0. To construct the t-statistic for this test, you can compute the following, either by hand or using SET or ANALYZ:[8]

$$t = (coef\ - a_0\ )/(std.\, err.)$$

If the HI option is used with OLSQ, the series $h_i = diag(X(X'X)^{-1}X')$ is stored under the name @HI. This series is useful for detecting "influential observations" (data errors, outliers, etc.). See the *Reference Manual* for details.

In the regression output, p-values are shown next to all test statistics produced. These p-values are the probability of seeing a test statistic at least this large when the null hypothesis is true. That is, small p-values imply rejection of the null hypothesis. With the REGOPT procedure, you can obtain additional regression diagnostics, such as: Breusch-Godfrey LM test for autocorrelation, Ljung-Box Q-statistics, ARCH(1) test, recursive residuals, Chow test for parameter stability, White or Breusch-Pagan heteroskedasticity tests, Shapiro-Wilk normality test, and Akaike and Schwarz Information Criteria (used for selecting lag length or sets of regressors). Try issuing a REGOPT (PVPRINT, STARS, LMLAGS=2, QLAGS=2, BPLIST=C) ALL; command before using OLSQ to see the range of available diagnostics. See the *Reference Manual* for further details.

```
                                 Equation   1
                                 ============

                 Method of estimation = Ordinary Least Squares


Dependent variable: CONS
Current sample:  1949 to 1975
Number of observations:  27

       Mean of dep. var. = 519.033     LM het. test = .783303 [.376]
   Std. dev. of dep. var. = 147.459    Durbin-Watson = .461739 [.000,.000]
Sum of squared residuals = 3958.17 Jarque-Bera test = 3.29859 [.192]
   Variance of residuals = 158.327  Ramsey's RESET2 = 16.7953 [.000]
Std. error of regression = 12.5828  F (zero slopes) = 3545.79 [.000]
               R-squared = .992999   Schwarz B.I.C. = 108.941
      Adjusted R-squared = .992719   Log likelihood = -105.645

           Estimated    Standard
Variable  Coefficient     Error       t-statistic   P-value
C         -17.8024      9.33496       -1.90707      [.068]
GNP       .633919       .010646       59.5465       [.000]
```

Figure 5.1 Ordinary Least Squares Output

[8] See Chapter 8 on hypothesis testing for more detail.

5.4 Two-stage least squares: 2SLS, INST

In the 2SLS statement, the dependent variable and independent variables are listed exactly as for OLSQ. However, they are preceded by the 2SLS options, which include the required list of instrumental variables. For example, in the illustrative model, government expenditures, G, the logs of the money supply, LM, and the trend variable, TIME, are considered as exogenous and may serve as instruments. The appropriate statement is

```
2SLS(INST=(C,G,TIME,LM)) CONS C,GNP;
```

Note that INST is an alias of 2SLS, so that the following command is identical to the previous:

```
INST(INST=(C,G,TIME,LM)) CONS C,GNP;
```

The constant C should always be used as an instrument. Two-stage least squares is 2SLS with all of the exogenous variables in the complete model (and no other variables) listed as instruments. Valid estimation can be based on a smaller number of instruments when the complete model involves many exogenous variables. Valid estimation can also be performed even when the rest of a simultaneous model is not fully specified. In these cases, the estimator is usually termed instrumental variables, rather than two-stage least squares. Of course, there must be at least as many instruments as the number of right-hand side variables.

The output of 2SLS for the consumption function in the illustrative model is shown in **Figure 5.2**. All statistics relating to residuals are calculated from the same formulas as OLSQ. The residuals are the "structural residuals",

$$e = \hat{y} - Xb$$

not the "second-stage residuals", which would be obtained by doing two-stage least squares literally in two stages (this would involve replacing some column(s) of X with predicted values from a first stage).

Here are the equations used by 2SLS to compute the estimated coefficients and their standard errors. Let ***Z*** be the matrix of values of the instruments, and ***y*** and ***X*** be the dependent and independent variables. The coefficient estimates are

$$b = [X'Z(Z'Z)^{-1}Z'X]^{-1}X'Z(Z'Z)^{-1}Z'y$$

The variance-covariance matrix of these coefficient estimates is given by

$$V(b) = s^2[X'Z(Z'Z)^{-1}Z'X]^{-1}$$

If the number of instruments ***Z*** is equal to the number of right-hand-side variables ***X***, the classical instrumental variable estimator results:

$$b = [Z'X]^{-1}Z'y$$

This follows easily from the previous formula for ***b***, since for this case

$$[X'Z(Z'Z)^{-1}Z'X]^{-1} = (Z'X)^{-1}(Z'Z)(X'Z)^{-1}$$

and four of the matrices cancel.

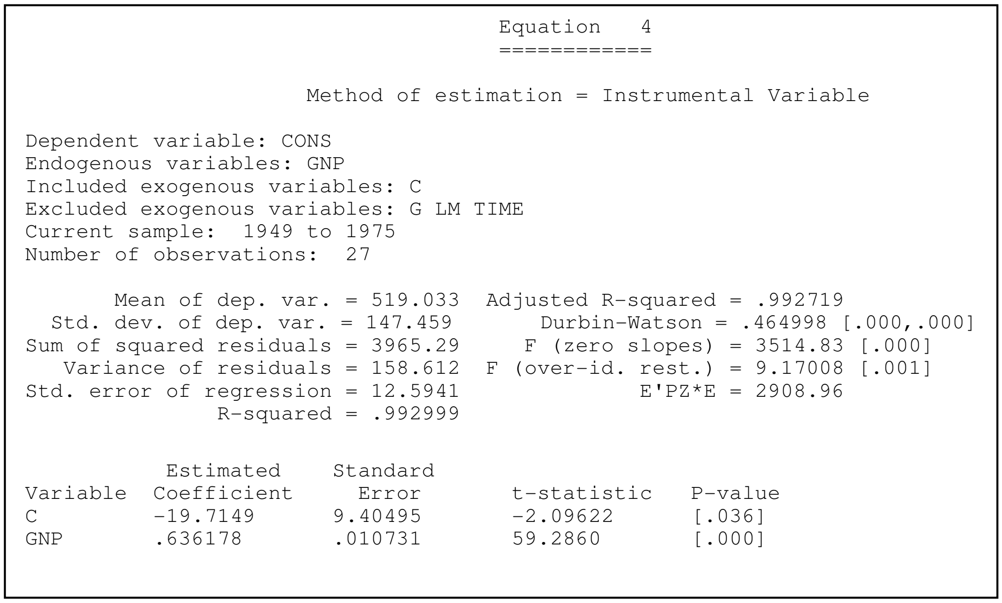
The objective function for 2SLS is not the ordinary sum of squared residuals, but rather the sum of squared residuals after projection onto the instruments (labeled E'PZ*E in the output):

$$e'P_Ze = e'Z(Z'Z)^{-1}Z'e$$

When the number of instruments (columns in ***Z***) is equal to the number of ***X***s (coefficients to be estimated), the model is exactly identified and ***Z'e*** will be zero. When the number of instruments is greater than the number of coefficients, a test for overidentification is available, defined as

$$F_{overident} = \frac{e'P_Z e}{(m-k)\hat{\sigma}^2}$$

where *m* is the number of instruments and *k* the number of coefficients. This test is approximately distributed as *F(m-k, N-k).*



```
                                   Equation   4
                                   ============

                      Method of estimation = Instrumental Variable

Dependent variable: CONS
Endogenous variables: GNP
Included exogenous variables: C
Excluded exogenous variables: G LM TIME
Current sample:  1949 to 1975
Number of observations:  27

       Mean of dep. var. = 519.033  Adjusted R-squared = .992719
  Std. dev. of dep. var. = 147.459       Durbin-Watson = .464998 [.000,.000]
Sum of squared residuals = 3965.29     F (zero slopes) = 3514.83 [.000]
   Variance of residuals = 158.612  F (over-id. rest.) = 9.17008 [.001]
Std. error of regression = 12.5941             E'PZ*E = 2908.96
               R-squared = .992999


           Estimated    Standard
Variable  Coefficient     Error       t-statistic   P-value
C         -19.7149      9.40495       -2.09622      [.036]
GNP       .636178       .010731       59.2860       [.000]
```

Figure 5.2 Two Stage Least Squares Output

5.5 *Limited information maximum likelihood: LIML*

The LIML command computes maximum likelihood estimates for a linear model with endogenous variables on the right-hand side (and normally distributed disturbances). It is specified like the 2SLS command with instruments given in the options list. The advantages of LIML over 2SLS are asymptotic efficiency and a small sample distribution with less bias. The FULLER option allows for additional small-sample corrections (see the *Reference Manual* or *Online Help System* for details). The disadvantage of LIML is that the estimates are more sensitive to specification error in the lists of included and excluded variables. (See Section 7.3 for estimation of nonlinear LIML models.) For the illustrative model, we have:

```
LIML(INST=(C,G,TIME,LM)) CONS C,GNP;
```

The output from this command is shown in **Figure 5.3**. The differences from the output for OLSQ and INST are the estimate of λ (the LIML eigenvalue), weak IV diagnostics (the estimated concentration parameter and the Cragg-Donald statistic), and the finite sample standard errors due to Bekker (1994).

In this example, both weak instrument tests are large and indicate that weak instruments are not likely to be a problem for this estimation. However the F-statistic for the overidentifying restrictions also suggest that the instruments are invalid. For more information on these statistics see the *TSP Reference Manual* and Hansen, Hausman, and Newey (2004).

```
                      Method of estimation = LIML

Dependent variable: CONS
Endogenous variables: GNP
Included exogenous variables: C
Excluded exogenous variables: G LM TIME
Current sample:  1949 to 1975
Number of observations:  27

       Mean of dep. var. = 519.033           R-squared = .992999
  Std. dev. of dep. var. = 147.459  Adjusted R-squared = .992719
Sum of squared residuals = 4063.75       Durbin-Watson = .465556 [.000,.000]
   Variance of residuals = 162.550  F (over-id. rest.) = 31.0720 [.000]
Std. error of regression = 12.7495  Lambda (eigenval.) = 3.70192
                   Concentration parameter =     1621.59
Cragg-Donald F-stat for Z2 in Reduced Form =     540.529

          Estimated    Standard
Variable  Coefficient     Error       t-statistic   P-value
C         -25.1644      10.0244       -2.51031      [.012]
GNP       .642613       .011477       55.9903       [.000]
Standard Errors computed from finite sample correction (Bekker)
```

Figure 5.3 LIML Output

5.6 First-order serial correlation: AR1

When the disturbances of a linear regression model are autocorrelated, the coefficient estimates of ordinary least squares are inefficient, and the standard error estimates are biased. The AR1 procedure provides several methods to obtain efficient estimates of an equation whose disturbances are autocorrelated of order one, that is, the observed disturbance is

$$u_t = \rho u_{t-1} + e_t$$

where e_t is homoskedastic and not serially correlated. For a discussion of the efficiency gains from the serial correlation correction and some Monte-Carlo evidence, see Rao and Griliches (1969). To estimate the consumption function in the illustrative model with a correction for serial correlation, replace OLSQ with AR1:

```
AR1 CONS C, GNP;
```

The default method of estimation is exact maximum likelihood. This method includes the first observation with a special weight, rather than simply dropping it, and imposes the stationarity requirement on the estimate of the serial correlation parameter by requiring the estimated ρ to be between -1 and 1. The small sample properties of this estimator, particularly its RMSE, are preferable to those of the conventional GLS procedures, unless there is a lagged dependent variable on the RHS.

When there is a lagged dependent variable, the “GLS" objective function is the default, because exact ML estimates will have a small sample bias. GLS drops the first observation, and is the same as nonlinear least squares on a rho-differenced equation. This method also can be requested for any model using the OBJFN option:

```
AR1 (OBJFN=GLS) CONS C GNP;
```

The GLS objective function is known to have multiple local optima in many cases, especially when there is a lagged dependent variable on the RHS, so a grid search is used by default to locate starting values, and then iterations are used to find the accurate global optimum.

A similar technique can be used to estimate higher order AR(p) (p>1) processes. The FORM(NAR=p) and LSQ commands make this easy (see Section 7.2.1 for more details on LSQ and the *Reference Manual* for details on FORM). Here is an example of AR(2) estimation (without a grid search to check for multiple local optima):

```
FORM (NAR=2) CAR2 CONS C GNP;
SMPL 51,75;       ? DROP FIRST 2 OBSERVATIONS
LSQ CAR2;
```

All these methods produce the usual regression output; the summary statistics are based on the ρ-transformed variables, while the plot displays the actual dependent variable (untransformed) and the corresponding fitted value, i.e.

$$\hat{y}_t = X_t b + \hat{\rho}(y_{t-1} - X_{t-1} b) = E(y_t \mid X_t, y_{t-1}, X_{t-1}, b, \hat{\rho})$$

Note that this is the same equation used to do a static forecast of the AR1 model (see Chapter 14). If the model is true, the residuals*,* y_t-$yhat_t$, will be uncorrelated. These actual and fitted values may be displayed in a plot with the

PLOTS option.

5.6.1 Instrumental variable estimation in AR1

Two-stage least squares and instrumental variable estimation with a serial correlation correction may be obtained using AR1 with the INST option, although only the GLS objective function is available. Several considerations in obtaining consistent estimates in this case are discussed by Fair (1970). To estimate the consumption function in our model by nonlinear two-stage least squares, we use

```
AR1(INST=(C,G,LM,TIME)) CONS C,GNP;
```

AR1 automatically adds lagged variables (in this example, CONS(-1) and GNP(-1)) to the list before estimating in order to ensure consistent estimation. To override this option and specify a complete list of instruments yourself, use the NOFAIR option:[9]

```
AR1(NOFAIR,INST=(C,G,LM,TIME,GNP(-1),CONS(-1))) CONS
        C,GNP;
```

Note that exactly the same model can be estimated by the LSQ command with instruments on a rho-differenced equation, like the example above with FORM.

5.7 Distributed lags

A distributed lag refers to the inclusion of more than one lag of a regressor (X variable) in a regression function. Many lags of a single variable may be easily included in the linear estimation procedures (OLSQ, 2SLS, LIML, AR1, PROBIT, TOBIT) of TSP. For example,

```
OLSQ CONS C GNP GNP(-1)-GNP(-15) ;
```

[9] This option is named after Ray Fair, who pointed out that the lagged dependent and independent variables should be in the instrument list to obtain consistent estimates. Fair retracted this claim in 1984 and it has since been disproved by Buse (1989). However, the alternate instruments required for consistency would involve pseudo-differencing with the estimated ρ, which is tedious to do by hand. Buse also showed that the most efficient estimator asymptotically in this case also includes the lagged excluded exogenous variables, but he cautions that in small samples this may quickly exhaust the degrees of freedom.

specifies a regression of consumption on GNP and 15 quarters of GNP lagged for the illustrative model. Note that OLSQ prints the Schwarz Bayesian Information Criterion and/or the Akaike Information Criterion, which may be used to select the length of an unconstrained distributed lag. A regression like this (with many unconstrained lags) frequently produces coefficient estimates that are highly imprecise and jump around from period to period. This has led econometricians to invent ways of smoothing the coefficient estimates. TSP supports two of these techniques: polynomial distributed lags, invented by Almon (1965) and Shiller lags, due to Shiller (1973).

5.7.1 Polynomial distributed lags

You can include PDL variables in any linear equation in TSP by including a variable or variables with a lag specification in parentheses in the list of independent variables. This lag specification has the form (n,p,constraint), where

n	Number of terms in the polynomial (degree plus one)
p	Number of lags in the lag distribution
constraint	Endpoint constraints: BOTH (2 constraints) FAR (1 constraint) NEAR (1 constraint) NONE (no constraints)

The number of parameters associated with the distributed lag is n less the number of constraints; this must be positive and less than or equal to p. Note that the number of lags includes the current observation, i.e., the lagged variables are (*t,t-1,t-2,..., t-p+1*).

Examples:

```
OLSQ CONS C,GNP(4,16,FAR) ;
```

specifies a regression of CONS on a distributed lag of GNP covering 16 quarters, having terms up to the third power, and obeying the far endpoint constraint. The output is shown in **Figure 5.4**.

```
OLSQ CONS C,GNP(4,16,FAR),R(4,24,NEAR) ;
```

specifies an additional distributed lag on R covering 24 quarters and a near endpoint constraint.

```
OLSQ CONS C,GNP(-1)(4,16,NONE);
```

forms a PDL of GNP that starts with the first lagged observation instead of the current observation.

When you use PDL variables in an instrument list, the choice of instruments depends on whether the variable is treated as exogenous or endogenous. If it is treated as exogenous, you want the scrambled variables described in the next section as instrumental variables. These could be specified by listing the names of the variables created by the "scrambling" process, i.e., in the above example, @PDL1, @PDL2, @PDL3. If PDL were treated as endogenous, it would suffice to ensure that there were enough instrumental variables in the list to meet the rank condition for identification -- at least as many as the number of variables in the regression after scrambling.

5.7.2 What PDL does

Each PDL term stands for a distributed lag of the form

$$b_0 x_t + b_1 x_{t-1} + \ldots + b_{p-1} x_{t-p+1}$$

The polynomial specification requires that b_i lie on a polynomial in i:

$$b_i = a_1 + a_2 \frac{(i+1)}{(p+1)} + \ldots + a_n \left[\frac{(i+1)}{(p+1)} \right]^{(n-1)}$$

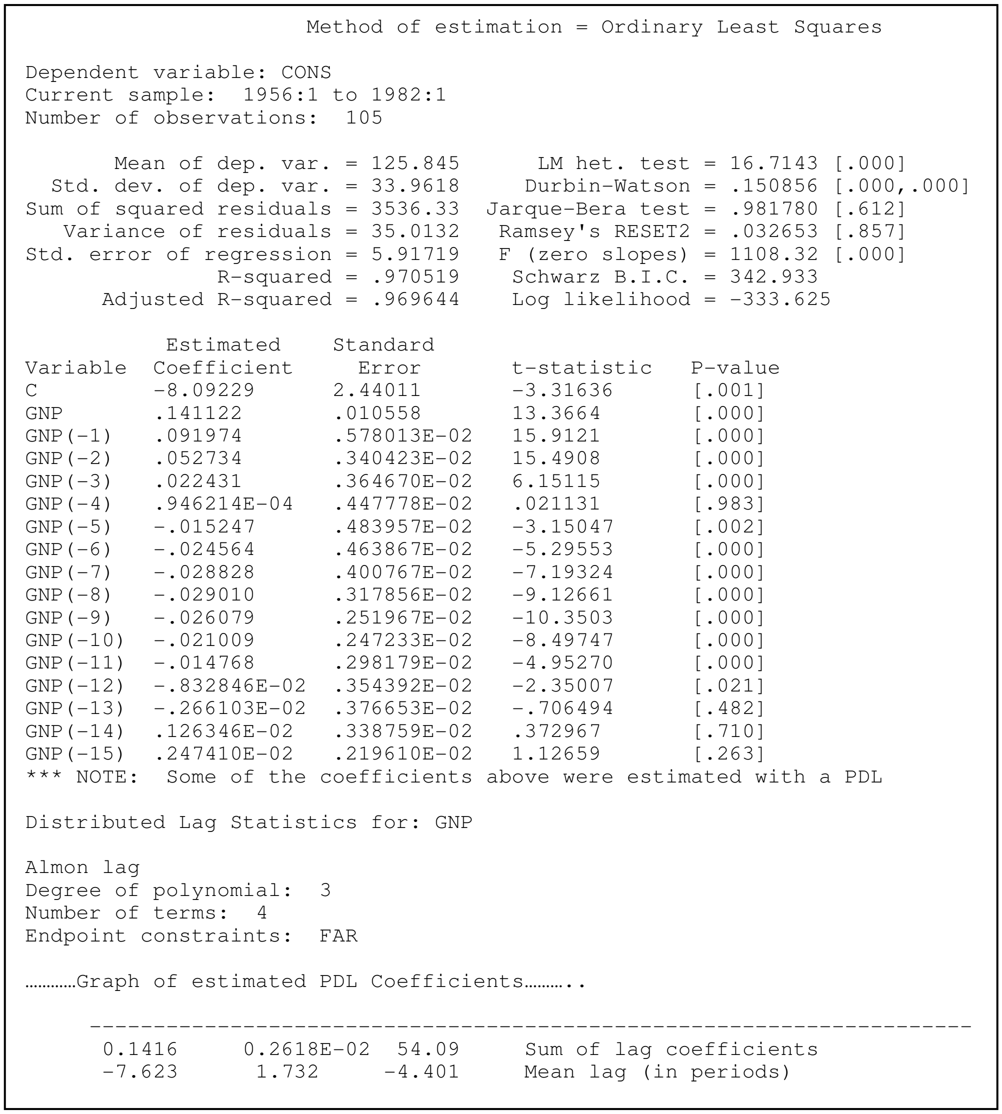
This implies constraints on the coefficients of x and its lags imposed by defining a new set of n variables z_t that are linear functions of the x variables. The actual functions differ slightly from what is implied by the above equations; see the Cooper (1972) article for details (Lagrangian interpolation polynomials are used). The variables $z_{1t},\ldots,z_{nt}$ enter the regression as ordinary independent variables (labeled @PDL1, @PDL2, and so forth in the regression results). Their coefficients are estimates of the parameters $a_1,\ldots,a_n$. After running this regression, TSP computes the corresponding estimates of the lag distribution coefficients $b_0, b_1,\ldots,b_{p-1}$ from the polynomial. The standard errors of the b_i are computed from the covariance matrix of the a_i using the delta method. In addition, the mean lag and the sum of the lags are presented along with their standard errors. The expressions defining them are

$$sumlag = \sum_{i=0}^{p-1} b_i = b_0 + b_1 + ... + b_{p-1}$$

$$meanlag = \frac{0 \cdot b_0 + 1 \cdot b_1 + ... + (p-1) b_{p-1}}{sumlag}$$

If the lag coefficients change sign, the mean lag computed by this formula has little meaning.

If endpoint constraints are imposed, the procedure is modified slightly. The near constraint sets a hypothetical b_{-1} to zero. TSP imposes this by dropping the first variable from the regression. The far constraint sets a hypothetical b_p to zero. TSP imposes this by subtracting the last variable from each of the others and dropping the last variable from the regression.



```
                      Method of estimation = Ordinary Least Squares

Dependent variable: CONS
Current sample:  1956:1 to 1982:1
Number of observations:  105

       Mean of dep. var. = 125.845      LM het. test = 16.7143 [.000]
  Std. dev. of dep. var. = 33.9618     Durbin-Watson = .150856 [.000,.000]
Sum of squared residuals = 3536.33  Jarque-Bera test = .981780 [.612]
   Variance of residuals = 35.0132   Ramsey's RESET2 = .032653 [.857]
Std. error of regression = 5.91719   F (zero slopes) = 1108.32 [.000]
               R-squared = .970519    Schwarz B.I.C. = 342.933
      Adjusted R-squared = .969644    Log likelihood = -333.625

           Estimated    Standard
Variable  Coefficient     Error       t-statistic   P-value
C         -8.09229      2.44011       -3.31636      [.001]
GNP       .141122       .010558       13.3664       [.000]
GNP(-1)   .091974       .578013E-02   15.9121       [.000]
GNP(-2)   .052734       .340423E-02   15.4908       [.000]
GNP(-3)   .022431       .364670E-02   6.15115       [.000]
GNP(-4)   .946214E-04   .447778E-02   .021131       [.983]
GNP(-5)   -.015247      .483957E-02   -3.15047      [.002]
GNP(-6)   -.024564      .463867E-02   -5.29553      [.000]
GNP(-7)   -.028828      .400767E-02   -7.19324      [.000]
GNP(-8)   -.029010      .317856E-02   -9.12661      [.000]
GNP(-9)   -.026079      .251967E-02   -10.3503      [.000]
GNP(-10)  -.021009      .247233E-02   -8.49747      [.000]
GNP(-11)  -.014768      .298179E-02   -4.95270      [.000]
GNP(-12)  -.832846E-02  .354392E-02   -2.35007      [.021]
GNP(-13)  -.266103E-02  .376653E-02   -.706494      [.482]
GNP(-14)  .126346E-02   .338759E-02   .372967       [.710]
GNP(-15)  .247410E-02   .219610E-02   1.12659       [.263]
*** NOTE:  Some of the coefficients above were estimated with a PDL

Distributed Lag Statistics for: GNP

Almon lag
Degree of polynomial:  3
Number of terms:  4
Endpoint constraints:  FAR

…………Graph of estimated PDL Coefficients………..

     ---------------------------------------------------------------------
      0.1416     0.2618E-02  54.09     Sum of lag coefficients
      -7.623      1.732     -4.401     Mean lag (in periods)
```

Figure 5.4 Sample PDL Output

5.7.3 Shiller lags

Constraining lag coefficients to lie exactly on a low-order polynomial often seems quite arbitrary, since there is no *a priori* reason for the functional form. The Shiller lag technique is a method for imposing smoothness on the lag coefficients without using an exact polynomial. The idea of the method is to impose a "smoothness" prior on the coefficients and estimate using Bayesian techniques. For a full discussion of the method, see Shiller (1973).

To use the Shiller lag procedure, you must specify a degree of differencing (this is similar to the polynomial degree specified in PDL), the number of lags of the X variable to include in the regression, the endpoint restrictions, and the value of the smoothness prior that you wish to use. For example, the command

```
OLSQ CONS C,GNP(3,16,NONE,.0025)
```

specifies a 2nd degree Shiller lag for GNP with no endpoint restrictions and a prior variance on the differenced coefficients equal to .0025. The output from this command is shown in **Figure 5.5**.

The prior variance on the differenced lag coefficients is what controls the smoothness of the distribution; a prior variance equal to zero yields PDL estimates, while a very large prior variance will give unconstrained lag coefficients. Other values yield answers between the smoothness of a polynomial and the jagged nature of unconstrained lags. One possibility is to use the sample variance of the unconstrained lag coefficients as a prior; this option is provided if you use a -1 as the prior variance.

One way to interpret Shiller lags is as a generalization of the PDL method. For example, the command

```
OLSQ CONS C,GNP(4,16,FAR,0)
```

estimates the same regression as the first example in section 4.7.1, since it specifies that the fourth degree differences of the coefficients should be precisely zero (with no error). This guarantees that all the coefficients will lie on a third degree polynomial, as for a PDL.

```
                      Method of estimation = Ordinary Least Squares


Dependent variable: CONS
Current sample:  1956:1 to 1982:1
Number of observations:  105

       Mean of dep. var. = 125.845      LM het. test = 15.7116 [.000]
  Std. dev. of dep. var. = 33.9618     Durbin-Watson = .153127 [.000,.000]
Sum of squared residuals = 3513.56  Jarque-Bera test = .873858 [.646]
   Variance of residuals = 39.9268   Ramsey's RESET2 = .080606 [.777]
Std. error of regression = 6.31877   F (zero slopes) = 182.272 [.000]
               R-squared = .970709    Schwarz B.I.C. = 372.845
      Adjusted R-squared = .965384    Log likelihood = -333.286

           Estimated    Standard
Variable  Coefficient     Error       t-statistic   P-value
C         -7.95317      2.62399       -3.03094      [.003]
GNP       .136131       .015219       8.94457       [.000]
GNP(-1)   .090098       .696157E-02   12.9421       [.000]
GNP(-2)   .052529       .527835E-02   9.95187       [.000]
GNP(-3)   .023264       .652659E-02   3.56451       [.001]
GNP(-4)   .179217E-02   .694711E-02   .257974       [.797]
GNP(-5)   -.012814      .647880E-02   -1.97786      [.051]
GNP(-6)   -.021813      .574975E-02   -3.79365      [.000]
GNP(-7)   -.026561      .538716E-02   -4.93038      [.000]
GNP(-8)   -.028239      .571193E-02   -4.94390      [.000]
GNP(-9)   -.027649      .653755E-02   -4.22921      [.000]
GNP(-10)  -.025167      .736512E-02   -3.41712      [.001]
GNP(-11)  -.020847      .761135E-02   -2.73889      [.007]
GNP(-12)  -.014557      .677608E-02   -2.14831      [.034]
GNP(-13)  -.612761E-02  .539660E-02   -1.13546      [.259]
GNP(-14)  .456186E-02   .832647E-02   .547874       [.585]
GNP(-15)  .017554       .017924       .979364       [.330]
*** NOTE:  Some of the coefficients above were estimated with a Shiller lag

Distributed Lag Statistics for: GNP
Shiller lag
Degree of polynomial:  2
Degree of differencing:  3
Prior smoothness std.dev. (xi):   0.25000E-02
Current smoothness std.dev.   :   0.65262E-03
Residual std.dev. used (sigma):    6.0999

----- Graph of estimated lag coefficients -----------
     ---------------------------------------------------------------------
      0.1422     0.2932E-02  48.48     Sum of lag coefficients
      -6.976      2.250     -3.101     Mean lag (in periods)
```

Figure 5.5 Shiller Lag Output

5.8 Weighted regression: the WEIGHT option

The weighted least squares estimator is useful when the variance of the disturbances in a regression differs across observations. It is most frequently used in cross-section regressions where the units of observation differ in scale or size -- for example, a cross section of states in the United States.

You can obtain weighted least squares in TSP by using the WEIGHT option in the linear regression procedures (OLSQ and INST). WEIGHT= should appear as an option in parentheses between the command name and the name of the dependent variable. After the = sign, put the name of a series whose values are proportional to the inverses of the variances of the disturbances in the regression. In cross section across political units, the weight variable should be the reciprocals of populations if the variables are sums or aggregates, and should equal population if the variables are measured per capita. For example, if YOUNG is the fraction of young adults living by themselves (a per capita variable) and POP is population, the following command estimates a regression using population values for the variables:

```
OLSQ (WEIGHT=POP) YOUNG,C,URBAN,
      CATHOLIC,ABORTION,SERVEMP,SOUTH ;
```

Two sets of summary statistics are reported, one based on weighted residuals computed by applying weighted least squares estimates to the weighted data, and and one based on unweighted residuals, computed by applying the weighted least squares estimates to the original data.

Note that both sets are based on the weighted least squares coefficients, which are minimum variance among all linear unbiased estimators if the weights are correct. The weighted residuals satisfy the usual properties of residuals and are the proper basis for statistical testing. The unweighted residuals describe the departure of the actual data from the regression function in their original units. TSP computes all the standard regression statistics for the weighted residuals and a subset of them for the unweighted residuals.

5.8.1 Normalization of weights

TSP divides the weights provided by the user by a constant so that the sum of the weights equals the number of observations. Observations given weight zero are not counted. This normalization does not affect the regression coefficients or most of the statistics. It leaves the magnitudes of the weighted data and weighted residuals the same, on average, as the unweighted data and residuals. Normalization may be suppressed with the UNNORM option. With UNNORM, if the weights are all large, the residuals, sum of squared residuals, residual variance, standard error, and implied number of observations will all be correspondingly large.

5.8.2 Weighted descriptive statistics

The WEIGHT= option may be used with any MSD statement to obtain weighted means, standard deviations, covariances, correlations, and moments. The data are multiplied by the square roots of the weight variable before the statistics are computed. The weights are normalized so that they sum to the number of observations; the UNNORM option is not available. Here is an example of using the WEIGHT= option:

```
MSD (WEIGHT=POP,CORR) YOUNG,C,URBAN,
       CATHOLIC,ABORTION,SERVEMP,SOUTH ;
```

5.9 Robust standard errors in the regression procedures

When you know the form and size of the heteroskedasticity in your data, you can use WEIGHT to obtain consistent estimates of the standard errors of your model. However, this is often not the case. The ROBUST option causes TSP to compute standard errors that are consistent even in the presence of unknown heteroskedasticity, by using the data to estimate its magnitude. Econometric references for this technique are White (1980a), White (1982), and Chamberlain (1984). Note that in general, White gives two terms in his formulae, the second of which vanishes if the model is correctly specified up to an additive error. TSP computes only the first term.

To obtain these heteroskedastic-consistent estimates of the standard errors of your model, include the option ROBUST on an OLSQ, 2SLS, or LSQ statement (see Chapter 7). For example,

```
OLSQ(ROBUST) CONS C GNP ;
```

computes the standard errors and variance estimate shown below for our illustrative example (compare to the conventional estimates in **Figure 5.1**).

```
          Estimated    Standard
Variable  Coefficient    Error       t-statistic   P-value
C         -17.8024      11.3535      -1.56800      [.129]
GNP       .633919       .013560      46.7507       [.000]
Standard Errors are heteroskedastic-consistent (HCTYPE=2).
```

For standard errors that are robust to simple autocorrelation, use the GMM(NMA=n) command. For example:

```
FORM HAC CONS C GNP;
```

```
GMM(HET,NMA=2,INST=(C,GNP)) HAC;
```

5.10 Quantile regressions: LAD

The ordinary least squares estimator is optimal when the disturbance in the equation is normally distributed. But when the disturbance is not normally distributed, other estimators are better. If the distribution is known, the efficient estimator is maximum likelihood with the correct distribution function. However, in many cases, you may suspect that your data distribution is "fat-tailed" or contains outliers, without knowing exactly its form. In this setting, the LAD estimator, which minimizes the sum of absolute deviations of the residuals, may be more efficient. The LAD estimator is also known as the L1 regression, least absolute residual (LAR), least absolute error (LAE), and minimum absolute deviation (MAD). See Chapter 22 of Judge et al (1988) for a discussion of its properties.

To estimate by least absolute deviations in TSP, use the LAD command like the OLSQ command. For example,

```
LAD CONS,C,GNP ;
```

Sample output from the LAD command is shown in **Figure 5.6** below. Note that standard errors are computed using bootstrap replications; the default number may be changed via the NBOOT option. See the Reference Manual for details on this and on the use of LAD for quantile regression and regression with a censored or truncated dependent variable.

```
                      Method of estimation = Least Absolute Deviations

Dependent variable: CONS
Current sample:  1949 to 1975
Number of observations:  27

       Mean of dep. var. = 519.033           R-squared = .992999
  Std. dev. of dep. var. = 147.459  Adjusted R-squared = .992719
Sum of squared residuals = 3992.70       Durbin-Watson = .465237
   Variance of residuals = 159.708      Schwarz B.I.C. = 108.643
Std. error of regression = 12.6376      Log likelihood = -105.347
  Sum of absolute residuals =   245.776
 Mean of absolute residuals =   9.10282
MSS / mod. Glejser het test =  0.336190     [.562]

                         Standard
Parameter  Estimate        Error       t-statistic   P-value
C          -20.6717      17.5460       -1.17814      [.250]
GNP        .638051       .023937       26.6559       [.000]

Standard Errors computed from  200 bootstrap replications
```

Figure 5.6 Sample LAD Output

6. MANIPULATION AND DISPLAY OF TSP VARIABLES

In the first few chapters of this manual, you learned how to read data into TSP, transform the data, and run simple linear regressions. Because TSP is a programming language tailored to the needs of econometricians, it can do much more than this. This chapter introduces some convenient features of the program for printing, plotting, sorting, and manipulating the data and regression results. It also describes a few procedures that perform certain specialized computations automatically, such as accumulating a capital stock from an investment series, Divisia indices, and seasonal adjustment.

To describe the full power of some of these procedures, we need to introduce the concept of a TSP variable type. You have already encountered the basic TSP type:

Series -- Time-series, cross-sectional or panel data. This variable is stored with both a frequency and starting date (or starting observation number, in the case of undated series). Observations outside the SMPL are treated as missing for the procedure being executed, even if they exist in data storage. Exceptions to this rule are lags and leads, which are obtained from outside the current SMPL. If you try to use a series whose frequency differs from the frequency specified by the current FREQ statement, TSP will give you an error message (unless you use CONVERT). Series are usually created with the READ, GENR and UNMAKE commands.

Besides the basic series variables, TSP allows the following kinds of variables:

Scalars -- These are variables that take on a single value. TSP distinguishes between two types of scalars: **Consts** and **Params**. **Consts** are ordinary scalars that are considered fixed or constant. They can be created by SET, CONST, UNMAKE, or even MATRIX statements, or as results from an estimation procedure. **Params** are a special type of scalar that are considered to be estimable coefficients in a TSP equation. They are declared (and optionally given starting values) in PARAM statements. Any **Param** can be made a **Const** (that is, can be held fixed for estimation) simply by changing its type with a CONST statement.

Just as GENR is used to do computations on time series, SET may be used to do computations on scalar variables and scalar elements of matrices and series. The rules for composing SET statements are the same as those for composing GENRs, except that the right-hand side must be a scalar-valued expression. Numbers or variable names in parentheses are used as subscripts in SET, while they would be interpreted as lags or leads in GENR. Such names are at most 4 characters for a single subscript, or 2 characters each for a double subscript (see Appendix A).[10] Some examples:

```
SET XMEAN = @MEAN(1) ;
SET XMAT(2,3) = 1.0 ;
SET XMAT(I,JJ) = Z(I)*W(JJ) ;
SET I1 = I-1;
SET XSIM(I) = XSIM(I1)*EXP(1.0+DELTA) ;
```

Matrices -- These include vectors as a special case and are described in more detail in Chapter 13. There are four kinds of matrices: general, symmetric, upper triangular, and diagonal. Matrices are created with the MMAKE, MFORM, READ, and MAT commands, and by many estimation procedures.

Lists -- Lists of TSP variable names. They are stored as the output of some procedures and may be created by using the LIST command. The name of a list can be used anywhere in TSP that a list of variable names can be used. Lists can be manipulated, changed, and even lagged. See the *Reference Manual* for details.

Equations -- Explicit (and often nonlinear) formulas for model estimation and simulation, usually created with FRML or IDENT statements (see Chapter 7).

[10] Subscript expressions like XSIM(I-1) are not allowed. (See the description of SET in the *Reference Manual*.) Two-dimensional matrices can have a single or double subscript (if it's a single subscript, the matrix is treated as a stacked series of columns). Scalars and subscripted vectors and matrices may be included in GENR statements also; they will be treated as though they have the same value for all observations, rather than being subject to the control of the current sample. Scalars and subscripted variables are legal syntax for command options and arguments that require a single value.

Equations are similar to GENR statements, except that they are not computed immediately when encountered; they are used only after the model is specified and the estimation or simulation command is invoked using their names as arguments. Equations may be printed, used for estimation and simulation, and computed by GENR or SET. Equations are also created from estimated linear models (FORM), and from operations on other equations (EQSUB and DIFFER).

6.1 Using the results of one procedure in another: COPY

Often, the computations you want to do depend on the results of a previous procedure. It can be tedious and time-consuming to run the first computations in one program and then enter those results by hand into a second program before proceeding. TSP provides a solution to this problem by automatically storing many of its most important results in data storage for use in later computations. They are given key names that always begin with the character @ so they will not be confused with the names of variables in your program.

For example, after every regression, two series are available in data storage: one called @FIT, the fitted values of the dependent variable, and one called @RES, the residuals (actual minus fitted values).[11] These series remain in data storage until they are replaced by another regression procedure. They can be used just like any other series before they are replaced. If you wish to save them, include a GENR or COPY statement after your regression:

```
OLSQ CONS C GNP ;
GENR CONSFIT = @FIT ; or COPY @FIT CONSFIT;
```

Other results you might want from a regression are: the estimated coefficients, stored as a vector called @COEF; the sum of squared residuals, stored as a scalar called @SSR; the standard error of the residuals, stored as a scalar called @S; and the variance covariance of the estimates, stored as a symmetric matrix called @VCOV. Almost all regression results are available in this form. See the *Reference Manual* for lists of the results available under each procedure.

Procedures other than the estimation procedures also store some of their results

[11] In multi-equation estimation with LSQ and FIML, @RES is stored as a matrix, but the column for a particular equation can be accessed easily with the UNMAKE command (see Chapter 13).

in data storage. For example, the MSD procedure stores the variable means, standard deviations, etc., in vectors named @MEAN, @STDDEV, etc.

COPY works with all TSP variable types except lists; use the LIST command to copy a list. For example:

```
LIST NEWLIST OLDLIST;
```

places the list defined by OLDLIST in storage under the name NEWLIST.

6.2 Printing series and other variables: PRINT, WRITE

WRITE and PRINT are synonyms. They are used to write or display any type of TSP variable; you can include different variable types in the same WRITE statement and each will be written using an appropriate format. The number of variables in one WRITE statement is limited only by the amount of working space and space available for the command line.

Here is an example showing the output format for the main TSP variable types. We print the items B1 (a scalar), CONSEQ (an equation), @VCOV (a symmetric matrix), and @RES (a series):

```
PRINT B1 CONSEQ @VCOV @RES;
```

Output from this example is shown in **Figure 6.1**.

When planning the format of your output, an important consideration is readability. Accordingly, when all the items are series, they are produced in table format with as many series per line as will conveniently fit on the page width specified for the current job. The first column of each table will contain the date or observation ID to label the observations. For items that are not series, each will be shown in a separate table. This gives you flexibility in arranging the tables in your output.

If any of the series or other variables have missing values, the number(s) in question will show a period (.) instead of a value.

```
B1 =    0.63392

EQUATION: CONSEQ

     FRML CONSEQ CONS = B0 + B1*GNP
                                           @VCOV

                      1            2
       1     307.86159
       2      -0.41535    0.00057296

                   @RES
1949          27.58024
1950          18.37168
1951          -4.86451
1952         -10.30162
1953         -11.86818
1954           0.00000
1955          -2.02385
1956           0.24342
1957           0.92301
1958           6.11628
1959           2.51999
1960           3.55599
1961           0.95206
1962          -6.29457
1963          -7.95699
1964          -8.53979
1965         -11.99944
1966         -19.15601
1967         -19.09194
1968         -17.02998
1969         -12.25735
1970           3.47583
1971           5.93063
1972           6.45060
1973           0.00000
1974           7.98380
1975          30.28837
```

Figure 6.1 Example of PRINT Output

6.3 Graphic displays of data

Frequently the best way to begin the analysis of a new set of data is by graphing it, either series by series or in more complex ways. TSP provides several routines for the graphical analysis of data: PLOT for one or more time series (assuming that the X axis is time), GRAPH for two variable plots (Y vs. X), and HIST for frequency distributions.

In most personal computer and unix versions of TSP, the plots produced by GRAPH and PLOT are displayed on the screen using high resolution graphics, and can be printed with software supplied with the program.

6.3.1 Plotting time series: PLOT, PLOTS, NOPLOT

PLOT produces a plot of one or more time series against time or against the observation number if the series is undated. For graphics plots, just specify the list of series you wish plotted (the formatting can be changed using Givewin's graphics editing features or options on the PLOT command). For character (paper) plots, the user specifies the series to plot and the character to use for each series in the plot. PLOT is followed by a series name, the character to use in plotting the series, possibly a second series name and a second character, and so on. Up to nine series may be plotted. The characters may be anything but $. ; ' " , . For example:

```
PLOT GNP CONS ;              ? graphics plot
PLOT GNP,*,CONS,X ;          ? character-based plot
```

Various parameters that control the appearance of the plot may be specified in an options list in parentheses following the word PLOT. These options are shown in the *Reference Manual*, although you may not need them if you like the appearance of the plot with the default settings. The default has a box around it but no vertical lines within the box, the maximum and minimum value are labeled, and the observations ID and series value are printed down the right and left side respectively.

For convenience, the PLOT options that you specify are retained in the next PLOT(s) until overridden either explicitly or by including the option RESTORE in the list. RESTORE resets the options at their default values.

Here are some additional examples of the PLOT command.

```
PLOT GNP GNPS CONS CONSS ;        ? Graphics versions
PLOT RESID ;
```

or

```
?  Character-based versions
PLOT(MIN=500,MAX=1500,LINES=(1000)) GNP G GNPS H CONS C
        CONSS D ;
PLOT(MIN=-25.,MAX=25.,BMEAN,HEADER,VALUES,BAND=STANDA
        RD,INTEGER) RESID * ;
```

There is also a command called PLOTS which "turns on" plots of residuals for all further estimation commands such as OLSQ, 2SLS, LIML, AR1, LSQ, FIML. The NOPLOT command "turns off" the residual plots. These automatic plots are useful for checking the fit of the model.

Figure 6.2 shows a plot produced by the Givewin version of TSP (using the Almon (1965) data on expenditures and appropriations).

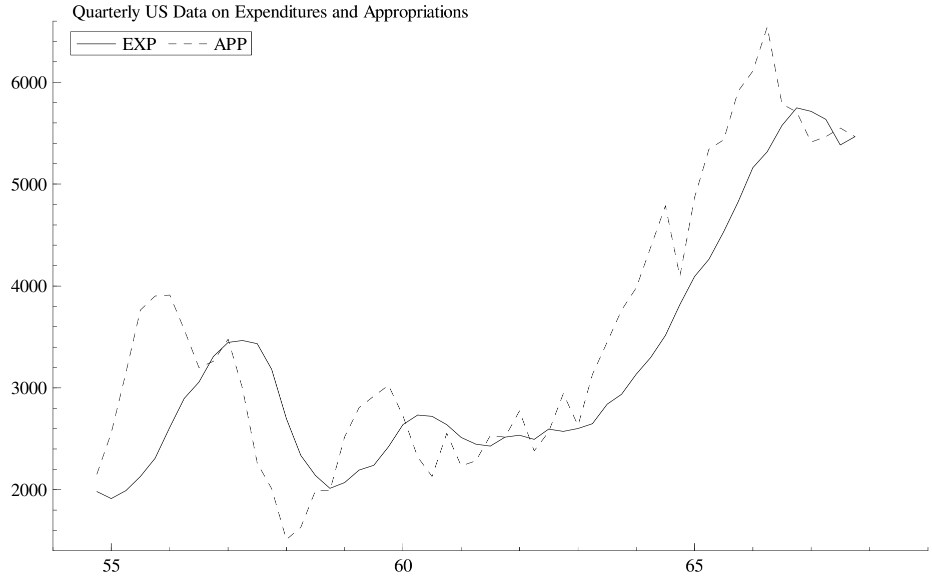


Figure 6.2 Sample Plot of Almon Data

6.3.2 Graphs or scatter plots: GRAPH

GRAPH produces a scatter plot of one series against another. As in the case of PLOT, GRAPH produces either high-resolution graphics (in most PC and unix versions) or printer character plots. List the x-axis series is first, and then as many y-axis series as you wish to include. For example, to obtain plots of actual and fitted values from a regression, use

```
OLSQ CONS C YD ;
GRAPH YD CONS @FIT ;
```

The above example produces a conventional graph of the actual and fitted values of CONS against the right-hand-side variable YD. The example below, from the illustrative model, generated the plot in **Figure 6.3**:

```
GRAPH (title="Compare Consumption Function Estimates")
      GNP CONS OLS LAD KERNEL ;
```

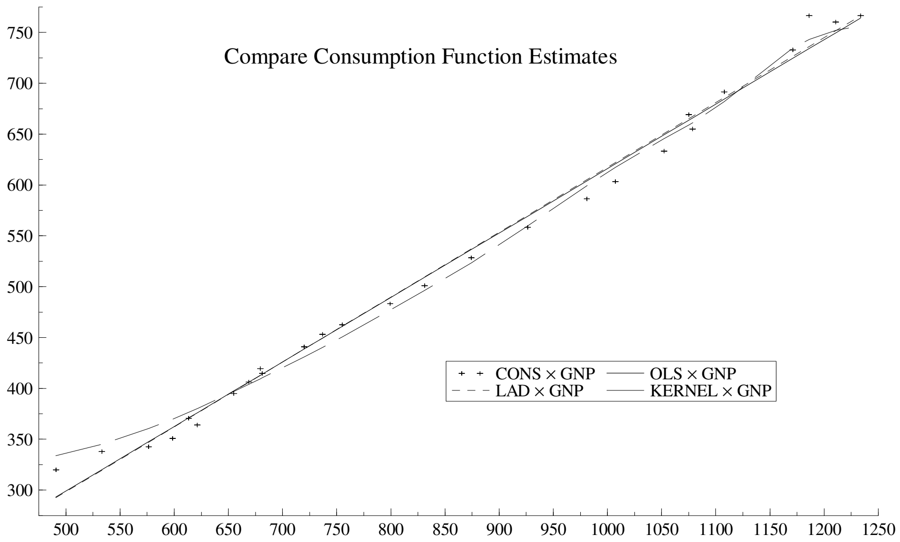


Figure 6.3 Graph of Consumption Function Estimates

6.3.3 Plotting histograms: HIST

HIST produces histograms (bar charts or frequency distributions) of series. It is convenient for obtaining a rough picture of the univariate distribution of your data. For example, from the illustrative model,

```
HIST (TITLE=”Distribution of short term interest rates’) RS;      ?
```

produces a bar chart with ten bars, indicating the number of observations distributed in sections along the range of the series RS. Figure 6.4 shows the resulting histogram:

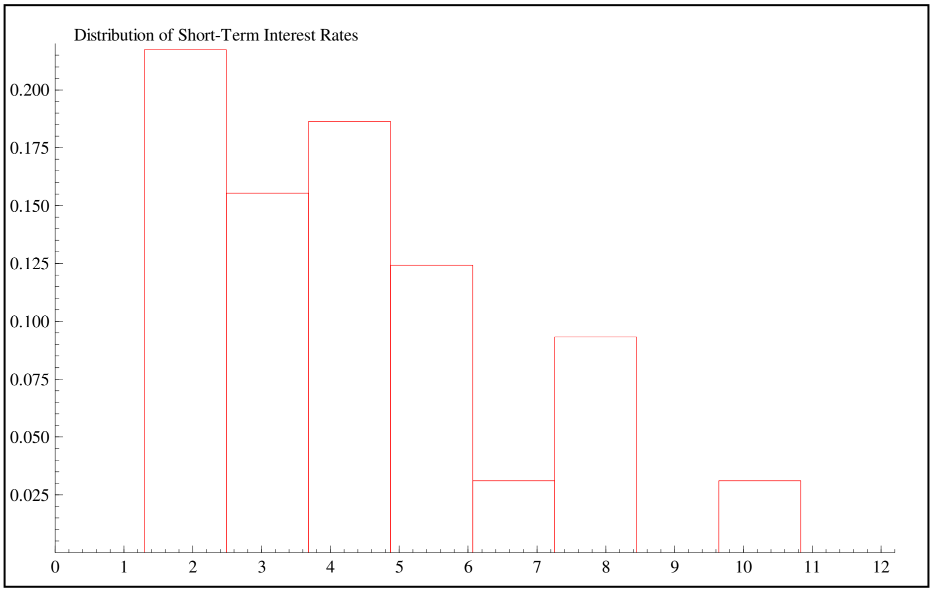


Figure 6.4 Sample HIST Output

Several options are available to control the number of bars, the range of the series, and the labeling and placement of the bars. These options described in the *Reference Manual*. For example,

```
HIST(DISCRETE) MERGYR;
```

produces a histogram of the variable MERGYR. Because the option DISCRETE was set, TSP creates a different bar for each unique value of MERGYR, on the assumption that it takes on only a small number of discrete values.

6.4 Sorting data: SORT

You can sort any individual series in ascending or descending order with the SORT command:

```
SORT X ; or SORT(REVERSE) X ;
```

You can also sort some or all of your series using a single series to determine the sort order followed by the list of the series to be sorted. For example, a set of panel data for ten countries, in which each country has three years of data (year 1 for all the countries, followed by year 2 for all the countries, etc.), could be reordered so that all the years for each country are adjacent:

```
KEY = COUNTRY*10+YEAR ;
SORT (ALL) KEY ;
```

The original order of the data was

COUNTRY	YEAR	KEY
1	1	11
2	1	21
.	.	.
10	1	101
1	2	12
2	2	22
.	.	.
10	2	102
1	3	13
2	3	23
.	.	.
10	3	103

and the new order will be

1	1	11
1	2	12
1	3	13
2	1	21
2	2	22
2	3	23
.	.	.
10	1	101
10	2	102
10	3	103

6.5 Dummy and trend variables: DUMMY, TREND

Dummy variables are series that are either zero or one in a given observation. They are typically used to measure things that are on or off (true or false). For example, the dependent variable of a Probit model is a dummy variable (see Section 9.2), and the constant term C in a regression is a dummy variable that is always "on". Dummy variables can be created in several ways. They can be LOADed directly, in which case a special convention for inputting repeated values (*) may be useful:

```
SMPL 1,20; READ D10; 10*1 10*0;
```

They can also be created in pieces under different SMPL statements. For example (also see TREND below):

```
SMPL 1,10; D10=1;
SMPL 11,20; D10=0;
```

Dummy variables can also be based on the values of other series. We have already seen examples of this in Section 3.5.1, where logical (true/false) expressions are used in a GENR command:

```
XPOS = X > 0 ;
```
or
```
GENR X1 = (X = 1);
```

Often the series in question takes on several distinct values or categories, and we would like to create a set of dummy variables (one for each category in the original series). This is easily done with the DUMMY command. For example, if the series EDUC takes the values 1, 2, and 3,

```
DUMMY EDUC;
```

would create the dummy variables EDUC1, EDUC2, and EDUC3. This would be equivalent to the commands:

```
EDUC1 = (EDUC=1); EDUC2 = (EDUC=2); EDUC3 = (EDUC=3);
```

DUMMY can easily create seasonal dummy variables. In fact, this is the default if the DUMMY; command is given (with no argument), and FREQ Q or M is in effect. The series Q1-Q4 or M1-M12 are created. Other DUMMY options are available to exclude the last dummy variable, and to save the names of the new variables in a TSP list. See the *Reference Manual* for details.

Another standard variable is the time trend, which typically equals 1, 2, 3, etc. This is created with TREND:

```
TREND T;
```

A trend variable is useful for estimating a deterministic trend component in a regression. It can also be useful for creating time-dependent dummy variables. The first dummy variable example above could have been created with:

```
SMPL 1,20; TREND T; D10 = (T <= 10);
```

The PERIOD= option in TREND allows the series to repeat with the specified periodicity. This is useful with balanced time-series/cross-section data, to make the trend start over at 1 for each individual. See the *Reference Manual* for details.

6.6 Computation of Capital Stock: CAPITL

In working with economic time series data, we are frequently given a gross investment series from which we wish to calculate the corresponding capital stock series based on an assumption about depreciation rates. The CAPITL command enables you to perform this computation easily in TSP. For example, if you have an annual investment series, I, and a depreciation rate of 10 percent per annum, you can compute a series of beginning-of-period capital stock, K, with the following command:

```
CAPITL I .10 K ;
```

The above CAPITL statement assumes that the value of the capital stock at the beginning of the SMPL was zero and that you wished to build it up from that point. Options are available to alter those assumptions. For example,

```
SMPL 58:1 83:2 ;
CAPITL (BENCHVAL=145.4,BENCHOBS=70:1) QINV .025 K ;
```

uses quarterly investment figures to compute capital stock, assuming that the first quarter of 1970 has a beginning period value of 145.4. Note that since this was quarterly data we used a depreciation rate of 0.025 to correspond to ten per cent per annum.

CAPITL calculates a capital stock series from a gross investment series, using a perpetual inventory and a constant rate of depreciation. Let I be gross investment, K be the capital stock, and δ be the rate of depreciation. Then CAPITL computes

$$K_t = (1-\delta)K_{t-1} + I_{t-1}$$

or

$$K_t = (1-\delta)K_{t-1} + I_t \qquad \textit{(if the END option is used).}$$

TSP starts from a capital stock benchmark at a specified observation. If the benchmark is in the middle of the sample as in our second example above, CAPITL also applies the reverse version of the formula,

$$K_t = (K_{t+1} - I_t)/(1-\delta)$$

or

$$K_t = (K_{t+1} - I_{t+1})/(1-\delta) \quad \textit{(if the END option is used).}$$

to compute values of the capital stock in periods before the benchmark.

A dynamic GENR statement is useful for simple explicit capital stock calculations. The first CAPITL example could be performed with the following statements:

```
SMPL 46,87; K=0;                  ? initialize capital stock to zero
SMPL 47,87; K = .9*K(-1)+I(-1); ? compute capital stock recursively
```

6.7 Divisia Indices: DIVIND

Another commonly needed manipulation of economic time series data is the computation of aggregate price indices from several underlying series. In TSP, you can use DIVIND to compute Divisia Indices, which have several desirable properties as index numbers:

1. They are chain-linked Laspeyres Indices, that is, for each year the current prices are used as a base in estimating the rate of growth to the following year.
2. They are also chained Paasche and Fisher Indices.

3. They are symmetric in prices and quantities.
4. If you reverse time and compute the indices backwards, you obtain the same result.
5. If you form indices of subgroups of the prices and quantities and then combine them using the Divisia method, the resulting index is the same as aggregating in one step from the original series.

These properties are contained in an unpublished note by W. M. Gorman (1970). For a more accessible discussion of their properties and a list of references, see Jorgenson and Griliches (1971) or Diewert (1976).

A Divisia index of prices is obtained by cumulating (over time) a weighted sum of the rates of change of the component prices. The weights are the current shares of the component goods in the total current expenditure on all the goods in the index. If we have *N* goods at time *t* with prices $p_{1t},...,p_{Nt}$ and quantities $q_{1t},...,q_{Nt}$, the rate of change of the logarithm of the Divisia price index p_t is given by

$$\log p_t - \log p_{t-1} = \sum_{i=1}^{N} w_{it}(\log p_{it} - \log p_{i,t-1})$$

The Divisia index of quantity can be obtained by applying the same strategy to quantities in place of prices, or, alternatively, by dividing total expenditure by the price index. However, the two quantity indices will not be exactly the same. If a quantity is zero or missing, that good is temporarily dropped from the index.

In TSP, the weights w_{it} used for calculating the index may be computed in three ways:

1. Arithmetic weights (ARITH option) are the arithmetic average of the expenditure shares ($p_{it}q_{it}/\Sigma p_{jt}q_{jt}$) in each of the two periods for which we are computing a rate of change.
2. Geometric weights (GEOM option) are the geometric average of the expenditure shares.
3. Combined weights (COMB option) combines the two weight computations given above, that is, they are a geometric average of the arithmetic weight, the share this period, and the share last period.

Here is an example of a Divisia command:

```
DIVIND (WEIGHT=ARITH,TYPE=P,PNORM=67) PIND, QIND, PS,
        QS, PND, QND, PD, QD ;
```

This command computes the indices PIND and QIND from the three pairs of inputs S, ND, and D. Price indices are computed and the quantity indices derived (TYPE=P). The index is normalized to one in 1967.

The default values of the DIVIND options are the following:

```
TYPE=Q, WEIGHT=COMB, QNORM=1, PNORM=1, NOPRINT,
        QVAL=1, PVAL=1
```

This means that quantity indices are to be computed using combined share weights, the first observation is to be normalized to have a quantity index of one, and the results are not to be printed. The computed indices will be stored in data storage under the names you gave the first two arguments. For further details on the options, see the *Reference Manual*.

6.8 Normalization of Series: NORMAL

NORMAL normalizes a series so that a chosen observation has an assigned value. It accomplishes this by dividing all observations of the series by the same number. After the word NORMAL, list the name of the series, the observation identifier of the base observation, and the value to be assigned. The normalized series will replace the original series. For example:

```
NORMAL CPI,75,100 ;
```

This is equivalent to the following statements:

```
SET CPI75 = CPI(75); GENR CPI = 100*CPI/CPI75;
```

6.9 Seasonal Adjustment: SAMA

TSP provides a simple moving average method for performing the seasonal adjustment of series. If seasonally adjusted data is very important in your work, you may wish to survey the literature in this area and make use of a more sophisticated or extensive program such as X-11 from the Census Bureau. A good place to start might be with Census Bureau (1976), in the references.

There are several seasonal adjustment methods other than the "ratio to a moving average". The proper method depends on the theoretical data generation process. Alternate methods include linear or log regression on seasonal dummies and inclusion of seasonal variables in the structural model. Seasonal dummy variables can be created easily with the DUMMY command. TSP provides the SAMA command because it would be difficult to compute with simple regressions; the other methods may be just as valid and they require little programming effort.

SAMA performs seasonal adjustment on quarterly or monthly time series. For example, suppose you have quarterly data on GNP from 1948 through 1982. You could seasonally adjust this series, store and print it with the following command:

```
FREQ Q ; SMPL 48:1 82:4 ;
SAMA(PRINT) GNPQ GNPQA ;
```

For details on the computations, see the *TSP Reference Manual.*

6.10 Principal Components: PRIN

Economic time series, particularly those for aggregate data, are frequently highly collinear; there is often very little information in a fourth or fifth series after you know the first three. Factor analysis with principal components can be a useful way of examining the similarities of data series. Principal components are a set of series constructed to explain as much variance of the original series as possible. Users of this procedure should be familiar with the method and uses of principal components, described in standard texts such as Harman (1976) and Theil (1971).

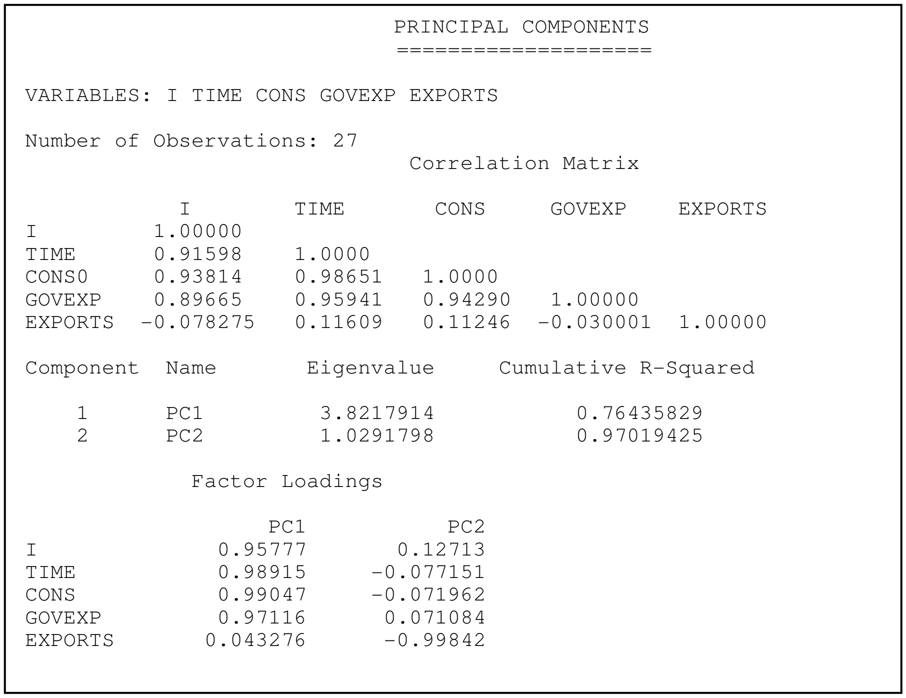
Here is an example of a PRIN command in TSP:

```
PRIN (NAME=PC,NCOM=3,FRAC=.95) I TIME CONS GOVEXP
        EXPORTS;
```

It specifies that three principal components are to be found of the five variables I, TIME, CONS, GOVEXP, and EXPORTS. If 95% of the variance of the five variables can be explained by fewer than three components, the program will stop there. The number of principal components actually constructed in any given procedure is the minimum of the number requested, the number needed to explain FRAC of the variance, and the number of series.

In the example given, only two principal components were actually constructed: the first was highly correlated with the level variables I, CONS, and GOVEXP, which in turn were highly correlated (over 95 per cent) with TIME. The second had negligible correlation with these variables and was almost entirely related to EXPORTS. Note that this example is illustrative only – this is probably not an appropriate way to analyze these data. The behavior of the first component suggests that is might be appropriate to check for unit roots and cointegration of the series.

In this example the principal components found will be stored under the names PC1, PC2, PC3, etc., for further use in the program. You may use any legal TSP name as the name for the principal components, but the names generated by adding the numbers must also be legal TSP names (that is, of length of less than 64 characters).



```
                             PRINCIPAL COMPONENTS
                             ====================

VARIABLES: I TIME CONS GOVEXP EXPORTS

Number of Observations: 27
                              Correlation Matrix

            I        TIME       CONS     GOVEXP    EXPORTS
I         1.00000
TIME      0.91598    1.0000
CONS0     0.93814    0.98651   1.0000
GOVEXP    0.89665    0.95941   0.94290   1.00000
EXPORTS  -0.078275   0.11609   0.11246  -0.030001  1.00000

Component  Name       Eigenvalue     Cumulative R-Squared

    1      PC1         3.8217914           0.76435829
    2      PC2         1.0291798           0.97019425

             Factor Loadings

                   PC1           PC2
I              0.95777       0.12713
TIME           0.98915     -0.077151
CONS           0.99047     -0.071962
GOVEXP         0.97116      0.071084
EXPORTS       0.043276      -0.99842
```

Figure 6.5 Sample PRIN Output

7. ESTIMATION OF NONLINEAR SYSTEMS OF EQUATIONS

In Chapter 5, you learned how to estimate various types of linear single equation models in TSP. These models were specified implicitly by listing the dependent variable and independent variables after the name of the estimation method (OLSQ, 2SLS, LIML, or AR1). Although this shorthand method of specifying a model is convenient, in order to estimate nonlinear models, you have to be more specific about the form of your equation(s).

Two procedures in TSP estimate general nonlinear models with additive disturbances: LSQ and FIML. LSQ is a minimum distance estimator that can be used to compute nonlinear single equation least squares, nonlinear two-stage least squares, nonlinear multivariate regression, SUR (seemingly unrelated regressions), nonlinear three-stage least squares, and some Generalized Method of Moments (GMM) estimators.

FIML obtains full information maximum likelihood estimates for a nonlinear simultaneous equation model whose disturbances are jointly normally distributed. For maximum likelihood on models with other distributions, use the general maximum likelihood (ML) procedure discussed in Chapter 9. Both LSQ and FIML can be used on linear as well as nonlinear models.

In this chapter we describe how to specify the equations of a model to be estimated and then discuss estimation with LSQ and FIML. In Chapter 8 we discuss hypothesis testing with applications for LSQ and FIML, and in Chapter 10 we describe the minimization techniques and convergence options used in all the nonlinear procedures.

7.1 Specifying the model: FRML, PARAM, etc

The first step in estimating a nonlinear model is to define the equations. The FRML statement is used to define TSP equations for estimation or other computations. To use FRML, you supply an equation in algebraic form (as in GENR) except that it is preceded by a name given to the equation when it is stored. Equations are referred to by their names when estimated and can be printed with the PRINT command.

For example, the FRMLs from the complete illustrative model at the end of

Chapter 3 are

```
FRML CONSEQ, CONS=A + B*GNP ;
FRML INVEQ, I=LAMBDA*I(-1) + ALPHA*GNP/(DELTA+R) ;
FRML INTRSTEQ, R=D + F*(LOG(GNP)+LP-LM) ;
FRML PRICEQ, LP = P(-1) + PSI*(LP(-1)-(LP(-2)) + PHI*LGNP +
        TREND*TIME + P0 ;
```

The rules for composing FRMLs are the same as for GENRs (see Chapter 3). The first FRML, CONSEQ, could be used to estimate a simple linear regression of CONS on GNP and a constant term by specifying two additional statements:

```
PARAM A,B; LSQ CONSEQ;
```

There is an implied additive disturbance term during estimation with LSQ or FIML. For example, the model estimated by the previous command is the following:

$$CONS_t = \alpha + \beta GNP_t + \varepsilon_t$$

If you are using simple linear equations, the FORM command can be used to choose parameter names automatically. For example, the command

```
FORM CONSEQ CONS C GNP;
```

would form the consumption equation and declare its parameters; it is equivalent to the two following statements:

```
FRML CONSEQ CONS = CONSEQ0 + CONSEQ1*GNP;
PARAM CONSEQ0 CONSEQ1;
```

FRMLs can also be used to specify functions of estimated parameters for testing (ANALYZ, Section 8.7), to specify formulas for equation substitution (EQSUB, Section 9.6) or differentiation (DIFFER, in the *Reference Manual*), and to specify log-likelihood equations (ML, Section 9.6). When used in this way, the equations do not have an implied additive disturbance.

Logical expressions may be included in FRMLs, but be aware that this will generally introduce a finite number of points at which your equation will not be differentiable. You can estimate or simulate with such an equation, but gradient methods may have difficulty converging if you happen to land on or near one of those points. The usual proofs of consistency, efficiency, and asymptotic normality will not go through in this case. However, such equations can be convenient when performing model simulations, and you may find them useful on occasion.

There are two ways to define a TSP equation: with the dependent variable on the left-hand side as in a GENR (a normalized equation), or as an unnormalized expression with no equal sign and no left-hand variable. An example of the latter would be

```
FRML CONSEQ2, CONS-A-B*GNP ;
```

This equation specifies the same model as the first version of CONSEQ, but as though it were written

$$\varepsilon_t = CONS_t - \alpha - \beta GNP_t$$

Unnormalized FRMLs are used by FIML, LSQ, GMM, and SIML to handle models that are nonlinear in the endogenous variables, for example, to specify an orthogonality condition for GMM estimation (see section 7.2.5). Here is an example of this kind of equation, where the dependent variable is a general expression:

$$y_t/q_t = \alpha + \varepsilon_t$$

The equation above should be defined as:

FRML EQR, Y/Q - A;[12]

The primary difference between using normalized and unnormalized equations in LSQ or FIML is that unnormalized equations do not have a well-defined left-hand side variable associated with them, so certain goodness-of-fit statistics such as R^2 cannot be computed.

ANALYZ and EQSUB also accept unnormalized equations. SOLVE will not accept unnormalized equations and returns an error message if it encounters

[12] If it were written as FRML EQR Y/Q = A; , the = sign would be interpreted as a logical operator:

```
FRML EQR, Y/Q .EQ. A ;
```

which means something entirely different from what was intended one.

A special form of FRML, called IDENT, is available to specify identities in a FIML model. These identities do not contain parameters to be estimated, nor do they have disturbances, but they are necessary to complete the model, that is, to insure that the model has a square Jacobian matrix (as many equations as there are endogenous variables). This condition is necessary for both FIML and SIML. (SIML actually treats FRMLs and IDENTs identically, but it may be useful to use IDENT statements instead of FRMLs for documentation purposes.) IDENT statements are given exactly like FRML statements except that they begin with the word IDENT instead of FRML. For example, the illustrative model is completed by a single identity relating GNP and consumption:

```
IDENT GNPID GNP=CONS+I+G ;
```

Information about the symbols in a FRML is provided in separate PARAM and CONST statements, for parameters to be estimated and constants, respectively. It is usually desirable to provide plausible numerical values for the parameters as starting points for estimations. Constants must be assigned values before estimation. For example, in the illustrative model (see Chapter 3),

```
PARAM A –18 B .62 LAMBDA .6 ALPHA 1 D –20 F 8 PSI .3 PHI .1
        TREND -.002 P0 -.6 ;
CONST DELTA 15 ;
```

The parameters need not be in order and may be defined in several PARAM and CONST statements. The only difference between a PARAM and a CONST is that PARAMs are estimated and CONSTs are not. Once parameters have been declared in a PARAM statement, they retain their estimability even if they are given new values by SET or UNMAKE commands. Parameters also retain their values from the most recent estimation. Parameters can be fixed at their current values by declaring them in a CONST statement. This will cause them to be treated as fixed until they are specified on a new PARAM statement.

7.2 Nonlinear least squares: LSQ

LSQ estimates single and multiple equation linear and nonlinear regression models. Depending on the number of equations and the specification of instrumental variables, several different econometric estimators are available with the LSQ command: nonlinear least squares, seemingly unrelated regression, nonlinear two stage least squares, three stage least squares, and generalized method of moments. Each estimator has a slightly different objective function which is minimized by means of iterative methods for nonlinear models. Details on the iterative techniques are in Chapter 10.

7.2.1 Single equation least squares

If LSQ is supplied with the name of just one FRML, and no instruments are specified, single equation least squares is the resulting estimator. For example,

```
LSQ CONSEQ;
```

estimates the consumption equation described earlier, assuming that A and B have been declared as PARAMs, and the series CONS and GNP have been properly defined over the current sample. In this example, the model is linear, so LSQ does not iterate, and the results are exactly the same as the command

```
OLSQ CONS C GNP;
```

OLSQ is easier to use for this model if you only want single equation estimation, but using LSQ automatically sets up starting values for a subsequent multiple equation estimation (which requires specifying the FRMLs).

For nonlinear least squares, the objective function is the sum of squared residuals (SSR). Minimizing SSR is equivalent to maximizing the likelihood function if the error in the equation is additive and normally distributed. The iteration technique is Gauss's method: derivatives of the equation residual with respect to each parameter are formed analytically. The current residual is regressed on the derivatives, and the resulting regression coefficients are the proposed changes in the parameters. It is possible to show that these coefficients will be zero if and only if the current parameters are at a local minimum of SSR. If the model was linear and the parameters had zero starting values, the first iteration would involve regressing the dependent variable (which equals the current residual) on the independent variables (which equal the derivatives of the residual with respect to the parameters, with a change of sign). For general nonlinear models, both the current residual and the derivatives will be functions of the current parameter values, but the procedure works the same, iterating until the derivatives are orthogonal to the residuals and the SSR is minimized.

A simple example of a nonlinear equation is the direct estimation of an AR1 model:

```
FRML CONSAR1 CONS = A + B*GNP + RHO*(CONS(-1) - (A +
        B*GNP(-1)));
PARAM A,B,RHO;
LSQ CONSAR1;
```

Note that this method of estimating an AR(1) model drops the first observation, like the AR1 procedure with the OBJFN=GLS option. Also see Section 5.6 for a more general and easier way to generate FRMLs for AR(p) estimation.

7.2.2 Multivariate regression and Seemingly Unrelated Regressions

If a model has two or more regression equations, it is likely that the disturbances from the two are correlated. If so, the technique of multivariate regression generally gives more efficient estimates than regression applied separately to each equation. Further, if two or more equations share the same parameter(s), they must be estimated jointly to impose these cross-equation constraints. This feature is particularly useful in estimating systems of demand equations derived from a utility function or a production function.

Multivariate regression is the simplest multiple equation estimator. It assumes there are no simultaneity problems with endogenous variables on the right-hand side of the equations. 3SLS and FIML are appropriate for joint estimation of equations with (or without) simultaneity; FIML will also estimate SUR models, often using less computer memory than LSQ.

An example of LSQ from the illustrative model (see Chapter 3) is

```
LSQ CONSEQ,INVEQ,INTRSTEQ,PRICEQ ;
```

This command specifies joint estimation of all four behavioral equations of the model. The output from this command is shown in **Figure 7.1**.

The multivariate least squares method is a generalized least squares method: the disturbances of the model are assumed to be independent across observations, but to have free covariance across equations. A consistent estimate of this covariance matrix is formed in some way or supplied to the procedure, and this estimate is used to weight the observations when the equations are re-estimated. The objective function can be written as

$$Q(b) = e(b)'(S^{-1} \otimes I_T)e(b)$$

where *e(b)* is the vector of stacked residuals (a function of the parameters *b*), *S* is an estimated covariance matrix of the disturbances and I_T is the identity matrix of order of the number of observations. If *S* is recomputed from $b^{(i)}$ at each iteration, this estimator is the same as the maximum likelihood estimator when the disturbances are multivariate normal.

Although any consistent estimator of *S* gives consistent parameter estimates for multivariate regression, the default method recomputes *S* from the estimated residuals at each iteration. This method has some desirable properties: in the case of consumer or factor demand systems, for example, this method yields estimates that are invariant with respect to which share equation is dropped.

If you want conventional Seemingly Unrelated Regressions estimates (two-step, rather than maximum likelihood), you can use SUR, which is an alias for LSQ with certain options preset. SUR obtains a consistent estimate of *S*, and then iterates only on *b* until convergence is obtained. More precise control over the initial estimate of *S* and over iteration on *S* is possible with the WNAME= and MAXITW= options (explained in detail in the *Reference Manual*).

```
EQUATIONS: CONSEQ INVEQ INTRSTEQ PRICEQ
MAXIMUM NUMBER OF ITERATIONS ON V-COV MATRIX OF RESIDUALS = 20
NOTE => The model is linear in the parameters.
Working space used: 3923
                                STARTING VALUES

        B0            B1         LAMBDA      ALPHA            D
VALUE   -17.80237   0.63392     0.69519    0.93262      -6.49935
       F           PSI           PHI        TREND            P0
VALUE  8.28104     0.96267    -0.036751  0.0020400       0.21353

F= 7.7250657561 FNEW= 7.5365117353 ISQZ= 1 STEP= 1. CRIT= 6.8450
------ iteration output -------
F= 7.2004917984 FNEW= 7.2004909522 ISQZ= 1 STEP= 1. CRIT= .25361E-04

CONVERGENCE ACHIEVED AFTER  17 ITERATIONS
   83 FUNCTION EVALUATIONS.
                          Residual Covariance Matrix
                CONSEQ         INVEQ      INTRSTEQ        PRICEQ
CONSEQ       148.65135
INVEQ        -69.32294     183.88108
INTRSTEQ       1.28343       0.93891       1.12461
PRICEQ         0.31574      -0.29112     0.0084148    0.00099038

                               Weighting Matrix
                CONSEQ         INVEQ      INTRSTEQ        PRICEQ
CONSEQ        0.082019      0.037881     -0.012755      -0.13640
INVEQ                       0.081230    -0.0096816      0.086872
INTRSTEQ                                   0.95436      -0.55702
PRICEQ                                                  85.55566

                  Covariance Matrix of Transformed Residual
       CONSEQ         INVEQ      INTRSTEQ        PRICEQ
CONSEQ        27.00000
INVEQ      3.86193D-15      27.00000
INTRSTEQ  -1.60733D-15  -3.71773D-16      27.00000
PRICEQ    -1.23344D-14  -1.17009D-14  -9.12410D-16      27.00000

Number of observations = 27       Log likelihood = -169.683
        Schwarz B.I.C. = 193.094

                             Standard
Parameter  Estimate           Error       t-statistic   P-value
B0         -23.3072      8.36041        -2.78780      [.005]
B1         .640115       .955276E-02    67.0084       [.000]
LAMBDA     .620197       .094824        6.54053       [.000]
ALPHA      1.15557       .281119        4.11060       [.000]
D          -6.37858      1.17706        -5.41906      [.000]
F          8.19093       .890868        9.19433       [.000]
PSI        -.456334      .128849        -3.54162      [.000]
PHI        .647412       .075012        8.63074       [.000]
TREND      -.019354      .259267E-02    -7.46476      [.000]
P0         -3.96640      .461670        -8.59142      [.000]

Standard Errors computed from quadratic form of analytic first
derivatives (Gauss)
----followed by equation by equation output
```

Figure 7.1 Sample LSQ (multivariate regression) Output

7.2.3 Nonlinear two-stage least squares: INST=

A nonlinear equation from a simultaneous model can be estimated by LSQ using a method developed by Amemiya (1974). The objective function for estimation is the sum of squared fitted residuals, where the fitted residuals are the fitted values from a regression of the true residuals on the instrumental variables. If the equation is linear in its parameters, this amounts to standard two-stage least squares, and LSQ will not iterate. If it is nonlinear, the estimates are consistent but not generally asymptotically efficient (relative to nonlinear three-stage least squares or FIML).

Nonlinear two-stage least squares in TSP is invoked by the INST= option. INST is followed by a list of instrumental variables in parentheses. For example, to estimate the investment function in the illustrative model of Chapter 3,

```
LSQ(INST=(C,G,LM,TIME)) INVEQ ;
```

7.2.4 Linear or nonlinear three-stage least squares: 3SLS

Three-stage least squares is an instrumental variable method for estimating a system of simultaneous equations where there may be endogenous variables on the right-hand side as well as contemporaneous correlation of the disturbances. The advantage of 3SLS over FIML is that the model does not have to be completely specified; the estimates for the equations and parameters can be consistent even if the exact form of the rest of the model is unknown. For example, you may have a set of equations describing the quantities demanded of certain goods as a function of the prices of goods. Prices may be determined as part of a larger economy that you do not wish to model explicitly. With the choice of suitable instruments, you could estimate the demand equations consistently without specifying the complete model. FIML would require specification of the price equations, although of course you could always do LIML using FIML by specifying a price equation that is a linear function of all the instruments.

To specify three-stage estimation use the INST= option and a list of equation names with 3SLS (or LSQ). For example, to estimate part of the illustrative model by three-stage least squares,

```
3SLS(INST=(C,G,LM,TIME)) CONSEQ,INTRSTEQ,PRICEQ;
```

The output from this example is shown on the following pages. (**Figure 7.2**)

The options for three-stage least squares are the same as those for univariate and multivariate regression. To request iteration over the covariance matrix of the residuals, use the MAXITW= option.[13] MAXITW=0 is the default.

3SLS estimates a set of equations by the same technique described for nonlinear two stage least squares, but considers the covariances across equations as well. The criterion for estimation is the sum of squared transformed fitted residuals. For each observation, fitted residuals are formed as the fitted values from regressions on instrumental variables. These are transformed by multiplying by the square root of the covariance matrix of the residuals. The contribution of the observation to the criterion is then the sum of squared values of these transformed fitted residuals.

For further details on the properties of this estimator, see Jorgenson and Laffont (1974) and Amemiya (1977). The NL3SLS estimator discussed by Amemiya is slightly more general in its choice of instruments than Jorgenson-Laffont; TSP uses the form specified by the latter where the same set of instruments is used for all equations. See GMM in the *Reference Manual* for two methods of specifying different instruments for each equation. The method of estimation is described in Berndt, Hall, Hall, and Hausman (1975). If the model is linear in its parameters and variables, three-stage least squares estimates are asymptotically efficient.

The three-stage estimation criterion requires an estimate of the residual covariance matrix. TSP obtains this by carrying out an initial estimation with the covariance matrix set equal to an identity matrix. If there are no parameters in common among the equations, these initial estimates are just the two-stage estimates. Then the covariance matrix is estimated from the true (not fitted) residuals from the initial estimates. Unless the user specifies otherwise, this estimate of the covariance matrix will be held fixed while the parameters are re-estimated to obtain three-stage least squares estimates.

[13] Note that iteration with MAXITW>0 may require a large number of iterations to converge. It is normally helpful only in demand systems to obtain estimates invariant to which share equation is dropped.

```
                           THREE STAGE LEAST SQUARES
                           =========================

EQUATIONS: CONSEQ INVEQ INTRSTEQ PRICEQ

INSTRUMENTS: C G LM TIME
MAXIMUM NUMBER OF ITERATIONS ON V-COV MATRIX OF RESIDUALS = 0

NOTE => The model is linear in the parameters.

Working space used: 4441

F= 3626.2758582  FNEW= 3561.9232085  ISQZ=  0 STEP= 1.    CRIT= 64.353
CONVERGENCE ACHIEVED AFTER   1 ITERATIONS
    2 FUNCTION EVALUATIONS.

END OF TWO STAGE LEAST SQUARES ITERATIONS (SIGMA=IDENTITY). THREE STAGE
LEAST SQUARES ESTIMATES WILL BE OBTAINED USING THIS ESTIMATE OF SIGMA:
                          RESIDUAL COVARIANCE MATRIX

                CONSEQ         INVEQ      INTRSTEQ        PRICEQ
CONSEQ       146.86262
INVEQ        -66.47420     185.26175
INTRSTEQ       1.42782      -0.25296       1.12389
PRICEQ       -0.077880       0.13425     0.0040156    0.00050308

F= 33.574061600  FNEW= 33.029434947  ISQZ=  0 STEP= 1.    CRIT= .54463
CONVERGENCE ACHIEVED AFTER   1 ITERATIONS
    4 FUNCTION EVALUATIONS.

--------continued on next page -----
```

```
                          THREE STAGE LEAST SQUARES
                          =========================

                         Residual Covariance Matrix
                CONSEQ          INVEQ      INTRSTEQ        PRICEQ
CONSEQ       147.57391
INVEQ        -67.94115      186.02693
INTRSTEQ       1.40383       -0.47859       1.12388
PRICEQ        -0.11551        0.17869     0.0033926    0.00064413

                             Weighting Matrix
                CONSEQ          INVEQ      INTRSTEQ        PRICEQ
CONSEQ        0.082517       0.036336     -0.010321      0.014659
INVEQ                        0.080277    -0.0024068     -0.032191
INTRSTEQ                                    0.94958      -0.20911
PRICEQ                                                   51.28644

                 Covariance Matrix of Transformed Residuals
                CONSEQ          INVEQ      INTRSTEQ        PRICEQ
CONSEQ        27.13077
INVEQ         -0.20479       26.92743
INTRSTEQ     -0.059263       -0.46166      27.04051
PRICEQ        -4.15961        3.11028      -0.26982      31.87350

Number of observations = 27  E'PZ*E = 33.0294

                         Standard
Parameter  Estimate        Error       t-statistic   P-value
B0         -21.5643      8.92636       -2.41580      [.016]
B1         .638235       .010190       62.6309       [.000]
LAMBDA     .708188       .157971       4.48301       [.000]
ALPHA      .893806       .466509       1.91595       [.055]
D          -6.57444      1.18326       -5.55619      [.000]
F          8.33843       .895622       9.31021       [.000]
PSI        1.12213       .319314       3.51421       [.000]
PHI        -.090421      .194599       -.464651      [.642]
TREND      .351033E-02   .673604E-02   .521127       [.602]
P0         .543581       1.18972       .456899       [.648]

Standard  Errors  computed  from  quadratic  form  of  analytic  first
derivatives (Gauss)

Equation: CONSEQ
Dependent variable: CONS

       Mean of dep. var. = 519.033
  Std. dev. of dep. var. = 147.459
Sum of squared residuals = 3984.50
   Variance of residuals = 147.574
Std. error of regression = 12.1480
               R-squared = .992999
           Durbin-Watson = .466537 [.000,.000]

------ followed by more equation by equation output ------
```

Figure 7.2 Sample Three Stage Least Squares Output

7.2.5 Generalized Method of Moments

3SLS coincides with the GMM estimator of Hansen (1982) when the errors are serially independent and the same instruments are used for each equation. In Hansen's notation, the GMM estimator sets the orthogonality conditions $u_t(b,y,X) \otimes z_t$ as close to zero as possible using the estimated variance of this vector as the metric. To perform this type of estimation in LSQ, define each element of the vector *u* as a normalized or unnormalized equation using FRML. The vector of *z*'s are specified as instruments in the INST list. The nonlinear three stage least squares estimates obtained are consistent and asymptotically efficient, and are also numerically identical to those obtained by the corresponding GMM estimator (for the default NOHET and NMA=0 options).

As an example, consider the simplest version of the Hansen-Singleton model [Hansen and Singleton (1982); for a simple presentation of the Euler equation for this type of model, see Hall (1978)]: a consumption-based asset pricing model where investors have a utility function of the constant relative risk aversion form. Denote consumption in period *t* as C_t and the one period return on asset *j* as x_{jt}. Then the representative agent model of intertemporal utility maximization implies the following population Euler equations in equilibrium:

$$E_t\left\{[\beta(C_t/C_{t-1})^{\alpha}x_{jt}-1]z_{mt}\right\}=0$$

where β is the discount rate and the z_{mt}, $m=1,...,M$ are in the agent's information set at time *t* (they may include such things as lagged asset prices and consumption). In Hansen's notation u_t is the expression in the square brackets and the *z*'s are the instruments.

This version of the Hansen-Singleton model can be easily estimated in the 3SLS procedure of TSP; the estimates coincide with GMM estimates, provided there is no serial correlation in the *u*'s (which will be true if the assets in *u* are stocks or other one period assets). Here is how to set up the problem when there are two assets and you wish to use four lags as instruments:

```
LIST LAGXS C X1(-1)-X1(-4) X2(-1)-X2(-4) ;
LIST UEQS U1EQ U2EQ ;
FRML U1EQ BETA*(CONS/CONS(-1))**ALPHA * X1(-1) - 1 ;
FRML U2EQ BETA*(CONS/CONS(-1))**ALPHA * X2(-1) - 1 ;
PARAM BETA 1 ALPHA -1 ;
GMM(HET,INST=LAGXS) UEQS ;
```

The preceding GMM example allows for conditional heteroskedasticity of the disturbances (the option HET, which is the default), but not for serial correlation. If the asset returns in the previous example were based on multi-period rather than one-period returns, there is no reason to expect that the covariance of the marginal utility of consumption with these returns will not be correlated across the periods comprising the multi-period returns (see Hansen and Singleton, section 2 for a further discussion of this point). In this case the estimates obtained by GMM will be consistent but not asymptotically efficient, since they use the "wrong" covariance matrix of the orthogonality conditions as a weighting matrix. For this case, the NMA option of the GMM command allows you to compute the correct weighting matrix automatically.

Suppose that in the previous problem you wanted to use a covariance estimate that incorporates moving average disturbances of second order. You would use the NMA option to specify this:

```
GMM(HET,NMA=2,INST=LAGXS) UEQS ;
```

Among others, Newey and West (1987) pointed out that the original estimate proposed by Hansen and Singleton in this case is frequently not positive definite in finite samples and proposed the use of declining weights to guarantee positive semi-definiteness. TSP offers two choices of spectral density kernels (KERNEL=BARTLETT or PARZEN) to compute these weights; the default choice is BARTLETT.

7.3 Full information maximum likelihood: FIML

FIML is the asymptotically efficient estimator for linear and nonlinear simultaneous models, under the assumption that the disturbances are multivariate normal. When this assumption fails, FIML may still be *asymptotically* efficient; see White (1982) or Gourieroux, Montfort, and Trognon (1984) for a discussion of when this will be true.

Because FIML operates on the model as a whole, the model must be complete -- it must have as many equations as endogenous variables. Thus in addition to the behavioral equations containing unknown parameters, FIML must be supplied with any identities that involve the endogenous variables. Identities provide a convenient way of entering repeated functions of endogenous variables into several equations; another way is the EQSUB command.

In FIML, the endogenous variables are listed in parentheses after the ENDOG= keyword. The corresponding instruments are then defined implicitly. For example, the illustrative model is estimated by FIML:

```
FIML(ENDOG=(CONS,I,R,LP,GNP))
        CONSEQ,INVEQ,INTRSTEQ,PRICEQ,GNPID;
```

The results of executing this FIML command are shown in **Figure 7.3** on the following pages.

The objective function for FIML is the log likelihood, which involves the log of the determinant of the residual covariance matrix, and the log of the determinant of the Jacobian (the derivatives of the residuals with respect to the endogenous variables).[14] If there no simultaneity and no nonlinear functions of the endogenous variables appear in the equations, the Jacobian term drops out, and the model is equivalent to multivariate regression.

FIML can be used to estimate nonlinear LIML models (see Section 5.5 for linear LIML models). In addition to the FRML for the equation of interest, FRMLs must be specified for each of the remaining endogenous variables. To make this a LIML model, these FRMLs should be linear in the instruments and there should be no constraints among their parameters. For example, to estimate the nonlinear AR1 equation of Section 7.2.1 with LIML:

```
FRML CONSAR1 CONS = A + B*GNP + RHO*(CONS(-1) -
        (A+B*GNP(-1));
PARAM A,B,RHO;
? Eq for GNP as function of the instruments:
FORM GNPEQ GNP C G LM TIME CONS(-1) GNP(-1);
LSQ(SILENT) CONSAR1 ;          ? Obtain starting values
LSQ(SILENT) GNPEQ ;            ? for parameters using LSQ
? Then estimate LIML model using FIML
FIML(ENDOG=(CONS,GNP)) CONSAR1,GNPEQ;
```

By default, the standard errors for the FIML structural parameters are computed from the matrix of sums of squares of the outer products of the gradient of the likelihood function with respect to both the structural

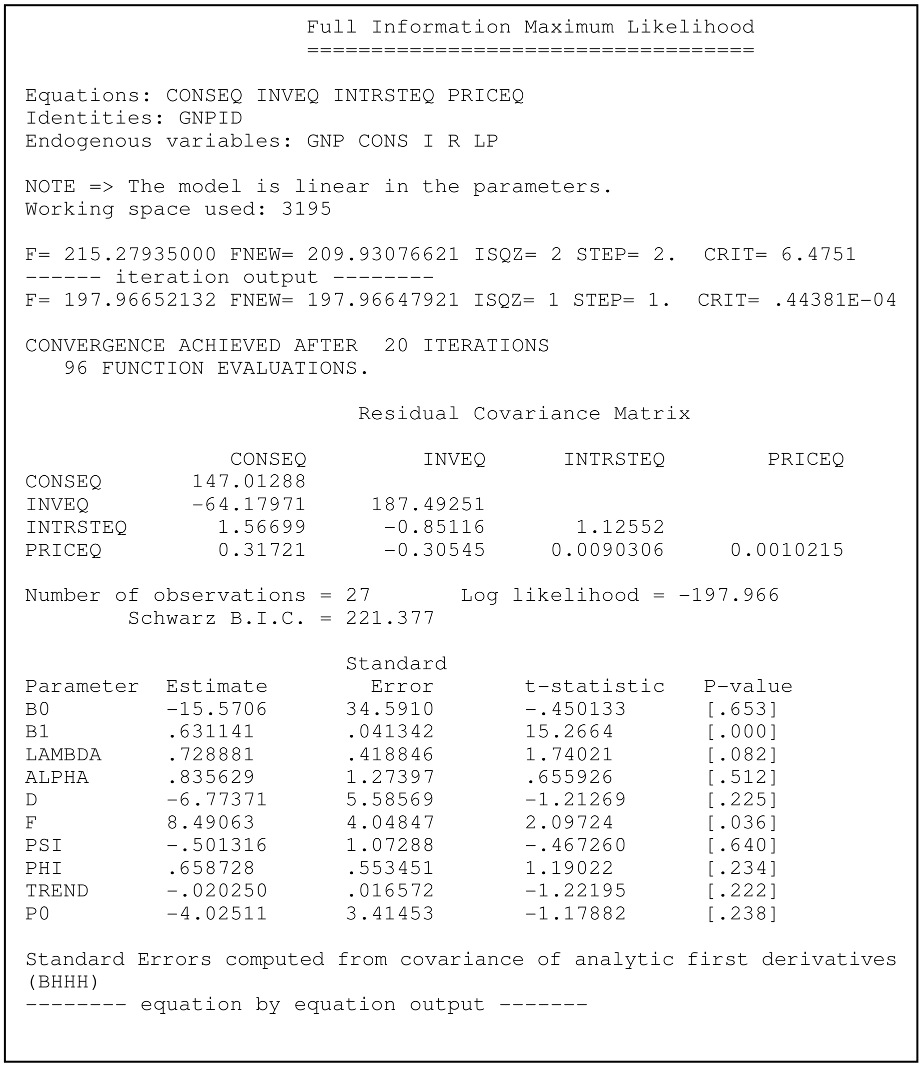
[14] Note that because all of TSP's nonlinear routines use minimization algorithms, the actual objective function used is the negative of the log likelihood. Final output for FIML and other procedures displays the correct value, however.

parameters and the unique elements of the inverse residual covariance matrix (the BHHH matrix). These standard errors are consistent and generally larger than those calculated in versions of TSP prior to 4.1, which were computed from the submatrix for the structural parameters only. For instance, the LIML example above has nine structural parameters and three covariance parameters (NEQ*(NEQ+1)/2).

Technical note: Calzolari and Panattoni (1988) studied eight alternate FIML standard error formulas and demonstrated the consistency and good small-sample performance of the FIML standard errors computed in TSP 4.1 and later versions.[15] They also showed that the R ("Gauss") matrix provided an inconsistent estimate of the standard errors when the model was nonlinear.[16] The R matrix is still used for iterations (HITER=G), but is not available for standard errors because of this inconsistency. Although HITER=C provides quadratic convergence close to the optimum and can be dramatically better than HITER=G, FIML uses HITER=G as the default, because Calzolari and Panattoni found it tends to perform better when the starting values are far from the optimum. HCOV=C (Hessian based on numerical second derivatives from analytic first derivatives) provides generally smaller standard errors than the default HCOV=B. Calzolari and Panattoni found that HCOV=B is usually closer to the true finite sample distribution of the parameters.

[15] The standard errors computed in TSP versions prior to 4.1 were not technically consistent, but have been shown to have reasonably good small-sample properties. TSP Version 4.1 was released in 1986, so it is unlikely you are using any version released prior to that date.

[16] This matrix was used by TSP 4.0 when the number of parameters was larger than the number of observations (in which case the BHHH matrix will be singular, since its rank will be less than its order).



```
                         Full Information Maximum Likelihood
                         ===================================

Equations: CONSEQ INVEQ INTRSTEQ PRICEQ
Identities: GNPID
Endogenous variables: GNP CONS I R LP

NOTE => The model is linear in the parameters.
Working space used: 3195

F= 215.27935000 FNEW= 209.93076621 ISQZ= 2 STEP= 2.  CRIT= 6.4751
------ iteration output --------
F= 197.96652132 FNEW= 197.96647921 ISQZ= 1 STEP= 1.  CRIT= .44381E-04

CONVERGENCE ACHIEVED AFTER  20 ITERATIONS
   96 FUNCTION EVALUATIONS.

                          Residual Covariance Matrix

                CONSEQ         INVEQ      INTRSTEQ        PRICEQ
CONSEQ       147.01288
INVEQ        -64.17971     187.49251
INTRSTEQ       1.56699      -0.85116       1.12552
PRICEQ         0.31721      -0.30545     0.0090306     0.0010215

Number of observations = 27       Log likelihood = -197.966
        Schwarz B.I.C. = 221.377

                          Standard
Parameter  Estimate         Error       t-statistic   P-value
B0         -15.5706      34.5910        -.450133      [.653]
B1         .631141       .041342        15.2664       [.000]
LAMBDA     .728881       .418846        1.74021       [.082]
ALPHA      .835629       1.27397        .655926       [.512]
D          -6.77371      5.58569        -1.21269      [.225]
F          8.49063       4.04847        2.09724       [.036]
PSI        -.501316      1.07288        -.467260      [.640]
PHI        .658728       .553451        1.19022       [.234]
TREND      -.020250      .016572        -1.22195      [.222]
P0         -4.02511      3.41453        -1.17882      [.238]

Standard Errors computed from covariance of analytic first derivatives
(BHHH)
-------- equation by equation output -------
```

Figure 7.3 Sample Output from FIML

8. TESTING HYPOTHESES

After you have estimated a model using one of the methods described in earlier chapters, you may want to test some hypotheses about the estimated parameters of the model. For example, you may want to test whether one or more variables belong in an equation, or whether the parameters satisfy some linear or nonlinear constraint. TSP has many methods for performing such tests, and in this chapter, we outline some of them; your ingenuity may find others.

Hypothesis testing of linear or nonlinear constraints on the parameters of econometric models can generally be performed using one of three methods. These methods are: the likelihood ratio (which compares constrained and unconstrained estimates), the Wald test (based on the unconstrained estimates), and the Lagrange multiplier, or LM, test (based on the constrained estimates).[17] In different situations, you may find one or the other of these tests easier to compute. Although all are asymptotically equivalent, in finite samples the results will differ (except in very specific simple cases). For example, in the case of linear constraints in a generalized least squares model, the three statistics obey the following inequality:

$$Wald \geq LR \geq LM$$

In this chapter we outline the tests that can be used with the linear estimation methods: the t-test, F-test, and a special version of the F-test called the Chow test. Then we discuss the tests available for testing nonlinear hypotheses, and linear or nonlinear hypotheses about nonlinear models. These fall into two basic classes, depending on the form of the hypothesis: the likelihood ratio and quasi-likelihood ratio test, and the Wald test. The t-test and F-test for linear models, and the Wald test for nonlinear models can all be done with the ANALYZ command. Finally we briefly discuss the Lagrange multiplier (score) test.

The LM test is an example of a type of test called a "specification test", for which a specific alternate hypothesis is often not available. Another large class of specification tests may be generated by a principle enunciated by Hausman (1977). The Hausman test is discussed in section 8.9 and an example is given in Section 13.4.1.

[17] See the excellent article by Engle (1985) for further information and references.

Classical hypothesis testing involves accepting or rejecting decisions based on tabulated significance levels for the theoretical distribution of the test statistic. TSP provides these tables in the form of a procedure (CDF) so that you can easily find the p-level for any test statistic you calculate. The Student's t, F, chi-squared, normal, and bivariate normal distributions are available. CDF also contains distributions for the Dickey-Fuller tests for unit roots and the Engle-Granger cointegration tests, which are discussed in Chapter 11.

8.1 t-tests

The simplest hypothesis test is the t-test, usually a test for equality of a single coefficient in your model with some prespecified value, the most common being zero. TSP always prints out the value of the statistic for the t-test of zero in the table of regression results.

In general, t-tests on single coefficients are of the form

(estimated coeff - hypothesized value)/(estimated std error)

For the standard case of testing against zero, this simplifies to the ratio of the estimated coefficient to its standard error. For example, if you want to test the hypothesis that the coefficient on X1 is one:

```
OLSQ Y C X1 X2 X3;
? T-test done manually:
SET TTEST = (@COEF(2)-1)/@SES(2);
? X1 is the second variable in @COEF and @SES
SET DFT = @NOB - @NCID;
? degrees of freedom - to compute the P-value
CDF(T,DF=DFT) TTEST;              ? print TTEST and its P-value
```

You can do the same thing more easily using the ANALYZ procedure (described below):

```
FRML TEST1 X1-1;          ? Hypothesize that X1=1
ANALYZ TEST1;             ? Compute F-test (1, .)
```

For this example, an even easier way to compute the test is to rewrite the model so that the hypothesis of interest is a t-test for a zero coefficient:

```
YX = Y-X1 ;     ? Impose the constraint that the coeff of X1 is 1
```

```
OLSQ YX C X1 X2 X3 ;
```

For this OLS estimation, the t-test for X1=0 is now a test that the coefficient in the original model was 1, and can be read off the regression output.

T-tests of the above kind, both simple and complex, may be performed on the results of any TSP linear or nonlinear estimation procedure. For some estimation methods, mostly nonlinear models, the resulting statistic will not be a t-statistic, but an asymptotically normal variable. The distribution of that statistic is almost the same as a two-tailed t-statistic based on more than 50 observations and provides a less conservative test in small samples. For certain time series regressions (those using integrated variables), tests like the above result in a Dickey-Fuller test, rather than a t-test.(See Section 11.6.)

T-tests can also be computed and displayed for estimators you have programmed with matrix procedures. Just supply the TSTATS command with the names of the coefficient vector and the variance-covariance matrix, and TSTATS will print a standard regression output table. For example:

```
TSTATS (NAMES=(BETA1-BETA7)) BETA VARB;
```

prints a table of the seven BETA coefficients, together with their standard errors (the square roots of the diagonal elements of VARB) and t-statistics for the hypothesis that each of them is zero.

8.2 F-tests

F-tests are commonly used to test linear hypotheses which involve more than one coefficient. The simplest way to compute F-tests is to use the ANALYZ command, where you write one FRML for each restriction. ANALYZ will compute the value of each restriction, the F-statistic for all the restrictions as a joint test, and the implied constrained values of the original coefficients.[18] For example, to test that the sum of the coefficients on X1 and X2 equals one, use the following commands:

```
OLSQ Y C X1 X2 X3;
? write the restriction so that the value would be zero
? if the restriction is true:
FRML SUM1 X1+X2 - 1;
```

[18] When used after a command other than OLSQ, ANALYZ computes an asymptotic chi-squared test.

```
ANALYZ SUM1;
```

Another way to test linear hypotheses is to estimate the unconstrained model, then impose the null hypothesis and estimate a constrained version of the model. In this case an F-statistic (with numerator degrees of freedom equal to the number of constraints and denominator degrees of freedom equal to the degrees of freedom in the unconstrained model) can be computed from the sum of squared residuals of the two models. Our example above had one constraint, that the sum of b_1 and b_2 was unity (where b_1 and b_2 are the coefficients of X1 and X2, respectively). We can write this constraint as b_1=(1-b_2), so we can impose this by regressing (Y-X1) on C, (X2-X1), and X3. The commands for this test are:

```
OLSQ Y C X1 X2 X3 ;        ? Unrestricted Model
SET SSRU = @SSR ;
SET DFU = @NOB-@NCID ;     ? Sum of squares and deg. of freedom
GENR YX1 = Y-X1 ;                 ? create variables
GENR DX = X2-X1 ;                 ? for restricted model.
OLSQ YX1 C DX X3 ;                ? run restricted model.
SET FSTAT = ((@SSR-SSRU)/1)/(SSRU/DFU) ;
CDF(F,DF1=1,DF2=DFU) FSTAT ;? Print out p-value for F-statistic.
```

TSP's internal variable @NOB contains the number of observations in the current SMPL and is always available for use in computations. @NCID is the number of estimated coefficients. Note the use of CDF to print out the significance level associated with this F-statistic.

One advantage of this last technique is that it easily encompasses situations where there may be many constraints, particularly situations consisting of several zero restrictions on variables. It is a variation on the quasi-likelihood ratio techniques discussed in the section on nonlinear two- and three-stage least squares.

8.3 Chow tests

Chow tests are a special form of F-test that check the stability of regression coefficients over two or more subsamples of the data. This is normally done by running a regression for the whole sample, then running the same regression for subsamples, and comparing the sums of squared residuals (SSRs). An F-test for the constraint that the two sets of coefficients are equal can be computed from the SSRs. However, TSP automates this particular form of the

F-test, using the options CHOW and CHOWDATE in the REGOPT procedure. Here is a simple example of a Chow test, both automated and manual:

```
SMPL 47:1 80:1;
REGOPT(CHOWDATE=60:2) CHOW;
? The two periods are 47:1-60:1; 60:2-80:1
OLSQ Y C X;  ? The Chow test is computed automatically.
```

To compute this F-statistic “by hand”, run the unconstrained and constrained regressions, and then form the F ratio:

```
? estimate for the whole sample.
SMPL 47:1 80:1 ;
SET DF = @NOB-4 ;
OLSQ Y C X ;
SET SSR0 = @SSR ;

? estimate for subperiod 1.
SMPL 47:1 60:1 ;
OLSQ Y C X ;
SET SSR1 = @SSR ;

? estimate for subperiod 2.
SMPL 60:2 80:1 ;
OLSQ Y C X ;
SET SSR2 = @SSR ;

? compute and print the Chow test.
SET CHOW = (SSR0-SSR1-SSR2)*DF/(2*(SSR1+SSR2)) ;
CDF (F,DF1=2,DF2=DF) CHOW ;
```

8.4 *Pseudo-F tests for 2SLS*

The 2SLS residuals are asymptotically normal, and can be used to form a pseudo-F test that has an approximate F distribution, following Startz (1983). The main difference is that the 2SLS objective function @PHI ($e'P_Ze$) is used in the numerator instead of @SSR. The previous F test example can be used to illustrate the pseudo-F test:

```
2SLS Y C X1 X2 X3 | C Z1-Z3 ;          ? Unrestricted model
SET PHIU = @PHI ; SET S2U = @S2 ;
SET DFU = @NOB-@NCID ;
GENR YX1 = Y-X1 ; GENR DX = X2-X1 ;
2SLS YX1 C DX X3 | C Z1-Z3 ;           ? Restricted model
SET PSEUDOF = ((@PHI-PHIU)/1)/S2U ;
CDF(F,DF1=1,DF2=DFU) PSEUDOF ;
```

Chow tests can also be performed with this methodology, but be careful to expand the instrument list for the restricted model (the one estimated on the whole period) by multiplying it by subsample dummies. This makes the instruments the same for all models being compared:

```
? set up the subsample dummies.
SMPL 47:1 80:1 ; LOW = 0 ; HIGH = 0 ;

? estimate for subperiod 1.
SMPL 47:1 60:1 ; LOW = 1 ;
2SLS Y C X | C Z ;
SET PHI1 = @PHI ; SET SSR1 = @SSR ;

? estimate for subperiod 2.
SMPL 60:2 80:1 ; HIGH = 1 ;
2SLS Y C X | C Z ;
SET PHI2 = @PHI ; SET SSR2 = @SSR ;

? estimate over the whole sample (restricted model)
SMPL 47:1 80:1 ;
ZLOW = Z*LOW ; ZHIGH = Z*HIGH ;
2SLS Y C X | LOW HIGH ZLOW ZHIGH ;
SET PHI0 = @PHI ;

? compute and print the pseudo-F version of the Chow test.
SET DF = @NOB-4 ;
SET PFCHOW = ((PHI0-PHI1-PHI2)/2) / ((SSR1+SSR2) / DF) ;
CDF (F,DF1=2,DF2=DF) PFCHOW ;
```

8.5 Likelihood ratio tests

In many ways, the likelihood ratio test is conceptually the easiest of the tests we consider in this chapter. It can be used for any model that is estimated by maximum likelihood. In TSP, this includes OLSQ, AR1, BJEST, LSQ without instrumental variables (nonlinear least squares or multivariate regression), FIML, PROBIT, TOBIT, SAMPSEL, LOGIT, and ML. For these methods, TSP prints out a value of the logarithm of the likelihood evaluated at the estimated parameters and stores it under the name @LOGL.

If L_1 is the value of the likelihood function for the maximum of the unconstrained model and L_0 is the value when the constraints are imposed, then the likelihood ratio test is computed as

$$LR = 2(L_1\text{-}L_0)$$

This test is always positive (or zero) since the likelihood of the unconstrained model is at least as high as that of the constrained model. The LR statistic is distributed asymptotically as a chi-squared variable with degrees of freedom equal to the number of constraints. For further information on the asymptotic properties of this statistic in nonlinear models estimated by maximum likelihood, see Gallant and Holly (1980).

To compute this test in TSP, save the two values of @LOGL for the unconstrained and constrained estimates, difference them, and multiply by 2. You can use CDF with the chi-squared distribution to obtain the associated p-value.

8.6 Nonlinear two- and three-stage least squares -- QLR

A quasi-likelihood ratio test can be used for hypothesis testing in a nonlinear model estimated by two- or three-stage least squares. This type of test is discussed in Gallant and Jorgenson (1979). The tests are based on the principle that if an estimated function of the data is asymptotically normally distributed with a variance-covariance matrix that can be consistently estimated, it is possible to construct a variable that is chi-squared distributed from this function of the data. In the cases discussed here, the function in question is the vector of differences between the residuals under the null and the residuals under the alternative, which has expectation zero under the null. The appropriate covariance matrix estimate is the minimum distance weighting matrix under the most unrestricted model.

The QLR test which Gallant and Jorgenson propose for three-stage least squares is the following:

$$T = n\ (Q_0 - Q_1)$$

where Q_0 is the value of the minimum distance criterion for the null hypothesis, Q_1 its value for the maintained hypothesis, and n the number of observations. The formula for the minimum distance function $Q(b)$, b a vector of parameters, is

$$Q = e(b)'(S^{-1} \otimes P_Z)e(b)$$

where $e(b)$ is the stacked vector of residuals from the model, S a consistent estimate of the covariance of the disturbances, and P_Z the projection matrix of the instruments, $Z(Z'Z)^{-1}Z'$. LSQ minimizes $nQ(b)$ when it is doing three-stage least squares and prints it as F =, FNEW =, and E'HH'E. It may be retrieved under the name @PHI after convergence.

A cautionary note on testing with a minimum distance criterion: the statistic can be ill-behaved (have the wrong sign) if the estimate of S is not held constant across the null and maintained hypotheses. To do so, obtain a consistent estimate of S using nonlinear two-stage least squares and then maintain that estimate as a constant weighting matrix while doing three-stage least squares on the two models. For example:

```
3SLS(INST=(Z1,Z2,...),MAXITW=0) MEQ1 MEQ2 ... ;
COPY @COVU S ;
3SLS(INST=(Z1,Z2,...),MAXITW=0,WNAME=S) EQ1 EQ2 ... ;
SET Q0 = @PHI ;
3SLS(INST=(Z1,Z2,...),MAXITW=0,WNAME=S) MEQ1 MEQ2 ... ;
SET Q1 = @PHI ;

SET TEST = Q0-Q1 ;
CDF (CHISQ,DF=# constraints) TEST ;
```

Q0 is the objective function for the null hypothesis and Q1 is the maintained hypothesis. The first estimator is nested within the second, hence Q0 ≥ Q1, and the test is necessarily positive. This test is asymptotically chi-squared with degrees of freedom equal to the number of restrictions imposed, that is, the number of parameters in the second model minus the number in the first.

For two-stage least squares the objective function used by TSP is

$$Q(b) = (1/n)e(b)'P_Z e(b)$$

In this case, the same kind of test can be computed as

$$T = n\ (Q_0 - Q_1)/s^2$$

where s^2 is a consistent estimate of the variance of the disturbance (in particular, one may use the standard error squared of the two stage estimate of the maintained hypothesis). In TSP, *nQ(b)* is stored as @PHI when two stage least squares estimates are obtained with LSQ, so that a QLR test for two-stage least squares can be done in the following way:

```
LSQ(INST=(Z1,Z2, ...)) MEQ1 ;
SET S2 = @S*@S ; SET Q1 = @PHI ;
LSQ(INST=(Z1,Z2, ...)) EQ1 ;
SET Q0 = @PHI ;
SET TEST = (Q0-Q1)/S2 ;
CDF (CHISQ,DF=# constraints) TEST ;
```

8.7 Wald tests for linear/nonlinear restrictions: ANALYZ

The likelihood ratio and quasi-likelihood ratio tests, which compare the estimates of a constrained and unconstrained version of the model, are best suited to testing hypotheses of the form

$$b = f(g)$$

that is, some larger set of parameters b is expressed as a set of nonlinear functions of a smaller set g. When the constraint is expressed this way, it is usually easy to write the equations for both the unconstrained and constrained models. However, likelihood ratio tests have the disadvantage of requiring the estimation of both forms of the model, which can become expensive (in terms of CPU time) for large nonlinear models.

The Wald test provides an alternate method for performing the same test. Asymptotically it is the same as the likelihood ratio test if the null hypothesis is true, although it may differ quite a bit in practice. Choosing among these tests is not straightforward. See Gallant and Holly (1980) and Berndt and Savin (1977) for further discussion.

In TSP, the Wald test can be done with ANALYZ; it is a generalization of the t- and F-tests described above. This test is also known as the delta method in nonlinear contexts. Suppose the hypothesis to be tested can be written as

$$h(b) = 0$$

where b is the vector of parameters of the unconstrained model and $h(b)$ is a set of m nonlinear constraints on those parameters. Given a set of estimates b and the associated covariance estimate $V(b)$, ANALYZ computes the constraints *h(b)* (a row vector) and their covariance matrix:

$$V[h(b)] = \left(\frac{\partial h}{\partial b}\right)' V(b) \left(\frac{\partial h}{\partial b}\right)$$

all evaluated at the estimated b vector. From *h(b)* and its variance we form a test statistic

$$T = h(b)V[h(b)]^{-1}h(b)'$$

This test statistic is distributed asymptotically as a chi-squared variable with degrees of freedom equal to *m* under the null hypothesis (when the constraints hold). The p-value of $\chi^2(m)$ is printed; if you are testing at the .05 significance level, and the p-value is less than .05, the null hypothesis is rejected.

ANALYZ computes this test if you specify the constraints *h(b)* as FRMLs in unnormalized form (no left-hand side variable or equal sign), estimate the unconstrained model, and then issue an ANALYZ command with the names of FRMLs as arguments.

Gallant and Jorgenson give an example of testing for symmetry in a translog consumer demand system with three goods (two equations to be estimated). The TSP commands to compute this example are:

```
FRML EQ1 Y1 = (A1 + B11*LNP1 + B12* LNP2 + B13*LNP3 +
        B1T*TIME)/
         ( -1 + (B11+B21+B31)* LNP1 + (B21+B22+B23)*LNP2
             + (B31+B32+B33)* LNP3 + (B1T+B2T+B3T)*TIME) ;
FRML EQ2 Y2 = (A2 + B21*LNP1 + B22* LNP2 + B23*LNP3 +
        B2T*TIME)/
         ( -1 + (B11+B21+B31)* LNP1 + (B21+B22+B23)*LNP2
             + (B31+B32+B33)* LNP3 + (B1T+B2T+B3T)*TIME) ;
PARAM A1 A2 B11-B13 B21-B23 B31-B33 B1T B2T B3T ;
LSQ (INST=(...list of instrumental variables...)) EQ1 EQ2 ;
FRML SYM1 B12-B21 ;
FRML SYM2 B13-B31 ;
FRML SYM3 B23-B32 ;
ANALYZ SYM1 SYM2 SYM3 ;
```

ANALYZ results consist of a table with the computed differences of the symmetry parameters and their implied standard errors. The value of the Wald test, degrees of freedom, and p-value follow.

8.8 Lagrange Multiplier Tests (Score Tests)

The Lagrange Multiplier test is generally based on the magnitude of the derivatives of the likelihood function with respect to the constraints evaluated at the constrained estimates. Sometimes you may find this test easier to compute, particularly when the alternate hypothesis is not well specified.

As Breusch and Pagan (1979) among others have shown, many LM tests can be computed as the number of observations times the R^2 from a particular regression depending on the hypothesis to be tested. For example, suppose you have estimated an equation (linear or nonlinear) under the assumption that the disturbances are homoskedastic and wish to test for heteroskedasticity as an unknown function of the exogenous variables. This unknown heteroskedasticity can be modeled as a polynomial function of the *X*s:

$$\sigma^2 = a_0 + a_1 X_{1t} + a_2 X_{2t} + a_{12} X_{1t} X_{2t} + a_{11} X_{1t}^2 + a_{22} X_{2t}^2 + ..$$

A test for heteroskedasticity of σ^2 is a test that $a_1=a_2=a_{12}=...=0$ in this equation. The test is performed by regressing a consistent estimate of σ^2 on the exogenous variables of the model and their powers, and computing the TR^2 of the regression, where T is the sample size:

```
OLSQ Y C X1 X2 ;
USQ = @RES*@RES ;
X1X2 = X1*X2 ; X1SQ = X1*X1 ; X2SQ = X2*X2 ;
OLSQ(SILENT) USQ C X1 X2 X1X2 X1SQ X2SQ ;
SET TRSQ = @NOB*@RSQ ;
CDF (CHISQ,DF=5) TRSQ ;
```

This particular LM test has been automated and is more easily computed using the WHITEHT option on REGOPT:

```
REGOPT(PVPRINT) WHITEHT;
OLSQ Y C X1 X2;
```

In the case of qualitative dependent variable models, the same method can be used, but with the derivative of the model with respect to σ^2 substituted for the dependent variable USQ. Here is a more complex example based on the sample selection model described in Section 9.3 of this manual:

Suppose that you wish to test for heteroskedasticity of the disturbance e_2 in the sample selection model, using the form

$$\sigma^2 = G(\alpha + X_t\gamma)$$

where G is an arbitrary monotonic transformation. Then (see Lee and Maddala 1985, Poirier and Ruud 1983) a simple LM test can be obtained by regressing the partial of the likelihood function with respect to σ^2 (evaluated at the maximum likelihood estimates obtained under homoskedasticity) on the X_t's and their powers and examining the TR^2 from that regression. In this case, the partial of the log likelihood function for a single observation with respect to σ^2 can be shown to be proportional to:

$$\omega = (e_{2t}/\sigma)^2 - 1 - \rho(e_{2t}/\sigma)\lambda_t$$

where λ_t is the inverse Mills ratio for the observation, stored by the SAMPSEL procedure as a series. The LM test for $\gamma = 0$ is a test for:

$$\frac{\partial \log L}{\partial \sigma^2} G'(a) X_t = 0$$

where the degrees of freedom for the test are the number of regressors in X_t and all quantities are evaluated at the maximum likelihood estimates obtained under the null hypothesis. Note that since $G'(\alpha)$ does not vary across observations, it drops out of the R^2 in the test, so that the exact monotonic transformation does not matter. The test is obtained by regressing ω on X_t and X_t^2 and examining the TR^2 from that regression. Here is an example of how to code this test in TSP:

```
SAMPSEL IN79 C LOGE76 | DLE7679 C LOGE76 ;
? List of names of sample selection coefficients:
LIST SSCOEF DO D1 BO B1 SIGHAT RHOHAT ;

UNMAKE @COEF SSCOEF ;
SMPLIF IN79 ;                      ? Choose observed data.
UHAT = @RES/SIGHAT ;               ? Standardized residuals.
DLOGL = UHAT*UHAT - 1 - RHOHAT*UHAT*@MILLS ;
LOGESQ = LOGE76*LOGE76 ;
OLSQ DLOGL C LOGE76 LOGESQ ;       ? Compute LM test.
SET TRSQ = @NOB*@RSQ ;
CDF (CHISQ,DF=2) TRSQ ;
```

8.9 Hausman Specification Tests

The Hausman specification test is based on the comparison of an estimator which is efficient (or more efficient) under the null hypothesis but inconsistent under the alternative with an estimator which is consistent (and less efficient) under both hypotheses. This principle may be applied in many settings; useful applications are testing for the exogeneity of instruments or testing for random versus fixed effects in panel data. When applied to the simultaneous equations setting to test for exogeneity, it is often called a Hausman-Wu test.

A general example of a Hausman test in TSP is given in Section 13.4.1, after we have presented matrix operations and matrices. Some classic applications of the Hausman test are tests for random versus fixed effects in panel data (now computed automatically in the PANEL procedure) and tests for the independence of irrelevant alternatives in the Logit model (Hausman-McFadden 1984), for which an example is given at the end of Chapter 9.

9. ESTIMATION OF QUALITATIVE DEPENDENT VARIABLE MODELS AND GENERAL ML ESTIMATION

Economists frequently encounter a research problem where the dependent variable of the structural model is not directly observed. For example, the actual value of the variable may be observed only part of the time and whether or not it is observed may depend on its value or on the values of other variables. Alternatively, you may observe a variable that takes on several discrete values related to an underlying unobserved dependent variable. This is the case, for example, when modeling the choice among alternatives such as different types of products or different modes of transportation. For these models, ordinary least squares or other standard econometric estimators are not appropriate, because of the limited or qualitative nature of the observed dependent variable.

The estimators described in this chapter are the following:

TOBIT -- The dependent variable is observed only when it lies above (or below) some threshold value.

PROBIT -- Only the sign (+/- or 1/0) of the dependent variable is observed, or the dependent variable only takes on two values (binary probit).

SAMPSEL -- The dependent variable is not observed when another unobserved variable in the model lies below a threshold value (this is a generalization of Tobit, where one equation is a Tobit equation and the other is a Probit).

LOGIT -- The dependent variable is the index of a choice among several discrete alternatives (e.g., 1, 2, or 3). There is a "value equation" for each alternative, and the one chosen has the highest value, although actual values are not observed. TSP will estimate models involving characteristics of the choice (conditional logit), the chooser (multinomial logit) or both (mixed logit).

ORDPROB -- The dependent variable is a choice among ordered discrete alternatives (usually 0,1,2,…). A set of ordered threshold values determine which category is chosen. This includes binary Probit as a special case.

POISSON -- The dependent variable is an integer count (0 or positive). The mean is an exponential function of the explanatory variables, and the variance equals the mean.

NEGBIN -- Same as Poisson, except the variance is proportional to the mean (NegBin 1), or a quadratic function of the mean (NegBin 2).

ML -- ML performs maximum likelihood estimation of any model for which the likelihood function can be written in a FRML, or evaluated in a PROC. It includes all of the above estimators as special cases.

The first seven procedures are designed for structural equations where the underlying dependent variable is linear in the parameters. Nonlinear structural equations can be specified for the models estimated by these procedures by writing the likelihood explicitly and using ML. All of the models are estimated by maximum likelihood, using common nonlinear maximization algorithms (see Chapter 10 for a discussion of nonlinear methods).

TOBIT, PROBIT, SAMPSEL, and ORDPROB models have normally distributed structural disturbances. The disturbances in the LOGIT model have the Generalized Extreme Value distribution (see Manski and McFadden (1981), Maddala (1983), or Train (1986) for further information about this distribution). Estimates obtained from the binary logit model are extremely close to those obtained by probit, up to the implied standard deviation of the disturbances (approximately 1.6 versus 1.0).

Standard errors robust to misspecification of the error term may be obtained for all models by including the HCOV=W or HCOV=NBW option. They are computed using the first and second derivatives of the likelihood function. However, be aware that recent Monte-Carlo work [Calzolari and Fiorentini (1990); Griffiths, Hill, and Pope (1987); Calzolari and Panattoni (1988)] has shown that neither the outer product (HCOV=B) nor the robust estimate (HCOV=W) are as accurate as the matrix of second derivatives (HCOV=N) in finite samples (approximately 100 observations) when the disturbances are truly normal. The outer product of the analytic gradients tends to overestimate the variance, and the robust estimate to underestimate it. For details, consult the articles cited.

9.1 TOBIT

The Tobit estimator was proposed by James Tobin (1958) when he was analyzing household expenditure on automobiles. The observed expenditure for many households was zero, implying that the desired quantity of automobile services was below the minimum price of a car. In this situation, estimation of a an expenditure model using ordinary least squares will result in coefficients biased toward zero. An alternative is the Tobit model, specified in the following way:

$$Y^* = Xb + \varepsilon \quad \textit{defines a latent variable } Y^*$$

$$Y = Y^* \qquad \textit{if } Y^* > 0$$

$$Y \textit{ not observed} \qquad \textit{if } Y^* \leq 0$$

For example, *Y** might be desired automobile expenditure and *Y* the actual expenditure observed.

TOBIT is specified like OLSQ, with the dependent variable and a list of independent variables. For example,

```
TOBIT CAREXP C,INCOME,MARSTAT;
```

In addition to the coefficient estimates, the output will include an estimate of SIGMA, the standard deviation of the disturbances ε. SIGMA is estimated jointly with the regression coefficients, implying that the standard error estimates for the coefficients and SIGMA are consistent estimates.

TOBIT assumes that any observations for which the dependent variable takes on a zero or negative value are observations not observed. You can easily estimate a model where the cutoff value is different from zero, or where observations with large values are those not observed, by using options. For example, if your dependent variable is Y, to use a cutoff value of THRESH other than zero, use

```
TOBIT (LOWER=THRESH) Y C X ;
```

If you want a ceiling for Y rather than a floor, then use

```
TOBIT (UPPER=THRESH) Y C X ;
```

TOBIT produces and stores the same output as OLSQ, except for some of the summary statistics (R-squared, Durbin-Watson, and F-statistic). The residuals for the nonmissing dependent variable observations are stored in @RES and may be printed or plotted. Values of @RES for the missing observations will be stored as missing.

See Maddala (1983), Chapter 6, and Amemiya (1986), Chapter 10, for further information on the Tobit model. The model estimated by the TOBIT procedure is Amemiya's Type I Tobit. Also see the *Reference Manual* for information on doing Tobit estimation with grouped data using the WEIGHT option.

9.2 PROBIT

Probit is used for analyzing the determinants of a choice between two discrete alternatives, such as working/not working. Since the dependent variable is not continuous, ordinary least squares is not appropriate. Instead, the dependent variable may be treated as an indicator of the sign of a latent continuous dependent variable. That is, the latent variable Y^* is given by the following regression equation, but only $Y\ (=0 \text{ or } 1)$ is observed:

$$Y^* = X\beta + \varepsilon$$

$$Y = 1 \text{ if } Y^* > 0$$

$$Y = 0 \text{ if } Y^* \leq 0$$

Often this latent variable has a meaningful interpretation, such as the net value of choice one (e.g., employment) versus choice zero (e.g., unemployment). Since the numerical scale of the latent variable is unobservable, the model is identified by normalizing the standard deviation (SIGMA) of the disturbance (ε) to one. The estimated coefficients in a Probit regression are not as easy to interpret has those for a linear or Tobit regression. For this reason, PROBIT also prints and stores a matrix, @DPDX, that contains the sample mean of the derivative of the probabilities $Y=0$ and $Y=1$ with respect to each independent variable.

Here is an example of a PROBIT command:

```
PROBIT WORKING C SCHOOL EXPER RURAL IND;
```

Output from PROBIT is similar to that for most of the regression procedures except that a quantity called “scaled R-squared” is also printed. This measure, due to Estrella (1996), is a generalized goodness of fit measure for a wide class of nonlinear models. It is computed by the following formula:

$$1-(\log L_u / \log L_c)^{-(2/N)\log L_c}$$

where $logL_u$ is the likelihood of the estimated model and $logL_c$ is the likelihood for a model with intercept only. This statistic has the familiar properties of an R-squared statistic: 1) it takes values on the unit interval with zero corresponding to no explanatory power and one to perfect explanatory power; 2) It is is based on the likelihood ratio statistic for the hypothesis that the coefficients of all the explanatory variables except the constant are zero; 3) Its derivative with respect to the test statistic is similar to that for the corresponding linear case; see Estrella (1996) for the details.

PROBIT can also estimate the selectivity bias in any kind of selection model, so that a correction can be computed for a two-stage estimator of such a model; in fact, PROBIT automatically stores a vector of estimated “Mills’ ratios” in @RMILLS. See the example in the SAMPSEL section below.

9.3 Sample Selection: SAMPSEL

The sample selection model is a generalization of the Tobit model when observability of the dependent variable (and possibly the independent variables) in the regression equation is affected by factors other than the value of the dependent variable. In the typology of Amemiya, this is a Type 2 Tobit model. In the original examples developed by Heckman (1974), Griliches, Hall, and Hausman (1978), Hanoch (1980), and others, the regression equation is the log of wages on schooling and experience. Obviously wages are only observed for people in the labor force, and labor force participation is governed by a separate probit equation describing the reservation wage. This equation is also called the selection equation; when its dependent variable is equal to one, the dependent variable in the regression equation is observed.

The sample selection model connects the two equations by estimating a correlation ρ between their disturbances. This correlation is implied, for example, when an unobserved characteristic of the individual enters both the wage and selection equations. When ρ is nonzero, estimation of the regression equation in the selected sample where it is observed would result in biased coefficients. To be precise, the model estimated by SAMPSEL is the following:

$$y_1 = X_1 b_1 + e_1 \qquad (y_1 \text{ not observed})$$

$$d_1 = 1 \text{ if } y_1 > 0$$

$$= 0 \text{ if } y_1 \leq 0$$

$$y_2 = X_2 b_2 + e_2 \qquad \text{if } d_1 = 1$$

$$\text{not observed} \qquad \text{if } d_1 = 0$$

$$Var(e_1) = 1 \; Var(e_2) = \sigma^2 \qquad Corr(e_1, e_2) = \rho$$

The selection (probit) equation and the regression equation are both specified on the SAMPSEL command, separated by a vertical bar (|), with the variables from the probit equation first. The wage and labor force participation example mentioned above would be specified as:

```
SAMPSEL INLABF C SCHOOL EXP MARST | LWAGE C SCHOOL EXP ;
```

where INLABF is a zero/one variable (d_1) and LWAGE is the logarithm of the wage (y_2).

The σ (SIGMA) and ρ (RHO) parameters are estimated jointly with the coefficients of both equations. SAMPSEL stores the fitted probabilities and implied inverse Mills ratio for the probit equation, as well as the residuals for the observed subset of observations. See the *Reference Manual* for further information about SAMPSEL's output.

Consistent estimates of the sample selection model can also be obtained with Heckman's two-stage method. This method involves estimating the selection equation with PROBIT, and using the inverse Mills ratio function of the probit residuals (stored by PROBIT in @MILLS) as an extra variable in a regression over the selected sample. For example, the example model described above could be estimated by the following commands:

```
PROBIT INLABF C SCHOOL EXP MARSTAT;
SMPLIF INLABF;
OLSQ(ROBUST) LWAGE C SCHOOL EXP @MILLS;
```

Unfortunately, the conventionally estimated standard errors for the coefficients obtained by OLSQ in this case are not consistent estimates. Better estimates can be obtained, however, by using the ROBUST option in OLSQ to compute heteroskedastic-consistent standard errors, although even these estimates do not correct for the fact that the inverse Mills' ratio contains estimated parameters. A second consideration is that the Heckman procedure is not fully efficient, because it is a two-step procedure with a non-diagonal information matrix. Therefore, the maximum likelihood procedure SAMPSEL will produce more efficient estimates (under the assumption of normal disturbances). See Section 8.8 for an example of testing for heteroskedasticity with SAMPSEL.

In practice, extra identifying variables in the selection equation are useful. Without them, the @MILLS term in the regression equation can be a proxy for left-out nonlinear functions of the right-hand-side variables. Accordingly, a correct functional form specification of the regression equation is quite important, as is the validity of the normality assumption. For information on estimating this model semi-parametrically, see the article by Powell in Volume IV of the *Handbook of Econometrics*.

It is also possible to use SAMPSEL to estimate models where only the selected sample is observed (the dependent variable in the selection model is always one for the observed data). See Bloom and Killingsworth (1985). However, identification of the parameters of the selection equation comes entirely from the implied weighting of the observed data by the normal cumulative distribution function, because there is no longer a separate probit equation in the likelihood function. In practice, this makes it difficult to obtain convergence.

9.4 Multinomial and conditional logit: LOGIT

When the dependent variable involves two or more discrete choices, the logit model can be a good way of examining the determinants of these choices. In the case of only two choices, it provides an alternative to the probit model; estimates from the two models will be very similar (see below). Logit models have different names and slightly different specifications, depending on whether the data or coefficients are choice-specific or chooser-specific. Multinomial logit has chooser-specific regressor variables and their coefficients vary over the choices. Conditional logit has choice-specific regressor variables and their coefficients are equal over all choices. Mixed logit involves both types of data and coefficients.

McFadden (1973, 1976, 1985) pioneered the use of the logit model in econometrics. The underlying model for most econometric applications involves latent value (utility) equations for each choice. For example, the following latent value equations apply to a model with three choices, with multinomial variable X that has coefficients $(\beta_1,\beta_2,\beta_3)$ and conditional variables (Z_1,Z_2,Z_3) that have coefficients γ:

$$V_1 = a_1 + X\beta_1 + Z_1\gamma + e_1$$

$$V_2 = a_2 + X\beta_2 + Z_2\gamma + e_2$$

$$V_3 = a_3 + X\beta_3 + Z_3\gamma + e_3$$

The latent values V_1, V_2, V_3 are not observed, but the chosen alternative is the one with the highest value. If alternative 2 is chosen, for example, we know that $V_2 > V_1$ and $V_2 > V_3$. If the disturbances e_1,e_2,e_3 have the Generalized Extreme Value distribution, the observed choice probabilities have the form

$$\Pr(i \ chosen \mid choice \ set \ J) = \frac{\exp(a_i + X\beta_i + Z_i\gamma)}{\sum_{j=1}^{J} \exp(a_j + X\beta_j + Z_j\gamma)}$$

In this case, the likelihood function is computationally simple, and the estimation method converges rapidly.

The LOGIT command resembles a regression command with the dependent variable followed by a list of the choice-specific (conditional) variables, and then a list of chooser-specific (multinomial) variables. The two lists are separated by a |. For example, the model outlined above is estimated with the command:

```
LOGIT(NCHOICE=3,COND) Y Z | C X ;
```

When multinomial variables are used, the model is identified by normalizing the multinomial coefficients of the first choice to zero. In the case of two alternatives, this provides compatibility with the PROBIT command, which effectively uses the same normalization; although the coefficients are larger in LOGIT by a factor of approximately 1.6, due to the larger standard deviation of the GEV disturbances.

Conditional logit models may have a variable number of choices per chooser, because it is the regressor variables that vary for each choice, while the coefficients are fixed across choices. When setting up data for this model, use one observation for each choice, instead of one observation per chooser. An additional variable, either case number, or number of choices, is used to keep the observations for each chooser together. See the *Reference Manual* for details. Remember that the constant C is not a legal conditional variable since it does not vary across choices. Choice-specific intercepts may be obtained by specifying C as a multinomial variable (after the |).

The basic logit model has a well-known shortcoming, the "Independence of Irrelevant Alternatives" property, sometimes called the "Blue Bus/Red Bus" problem. That is, in this model the ratio of probabilities between any two choices is unaffected by the availability of a third choice. This property derives from the independence assumption on the e_i that is necessary to generate the logit probabilities. Hausman and McFadden (1984) offer a specification test for the consistency of logit estimates with the IIA property. If this test rejects, there are two possible solutions to this problem within the logit framework; the alternative is to use a more flexible joint distribution for the $e_i s$ that allows correlation among them such as the multivariate normal distribution, although this is more difficult to implement.[19]

The nested logit model is another way to structure the alternatives so that some are "closer" to each other than others. Although this model is not currently implemented in TSP, it can be consistently estimated by using the conventional LOGIT procedure in stages. The first stage would estimate the bottom branch of a set of choices, and the succeeding stages would include a predicted "inclusive value" based on the estimates of lower stages in the set of independent variables. See Train (1986) or Manski and McFadden (1982) for details. Also see Section 9.6.10 for an example of how to estimate the nested logit model directly using the ML procedure and the TSP examples on the web site for a more complex nested logit example.

Another method to avoid the IIA problem within the logit framework uses the fact that logit models can be used to describe *any* set of choice probabilities, not just those implied by the utility model and the GEV distribution (McFadden 1975). In particular, to allow correlation among choices, include variables describing other alternatives in the latent value function for one alternative. The defect of this approach is that it does not specify how many or which variables should be added, and therefore may require estimation of a large number of parameters. Having a model of the possible relationships among choices will help here.

[19] See the advanced examples on the TSP website for multivariate probit estimation, and example 9.7.6 in this chapter for bivariate probit.

9.5 Estimation with an Ordered Dependent Variable

In some models, the available outcomes have a natural ordering, such as when a person has a low (coded as 0), moderate (1) or high (2) income. If the boundaries between these categories are known, or the categories have a natural scale, it could be estimated with a series of one or two-limit Tobit expressions. If the outcomes have natural units, such as the number of doctor visits, or the number of patents, then count data models are appropriate (see the next section). The common feature of the models in this section is that there is a single latent variable and hence only one disturbance, although there are multiple possible outcomes. This is in contrast to the multivariate or multinomial logit considered earlier and arises from the fact that the data have a natural ordering.

9.5.1 Ordered Probit: ORDPROB, INTERVAL

Ordered Probit handles the case where the (ordered) boundary values are unknown, and must be estimated. INTERVAL is just like Ordered Probit, but where the boundary values are known. In this case they must be supplied to the procedure using the LOWER and UPPER options; see the *Reference Manual* for details on using INTERVAL.

In the model estimated by the ORDPROB procedure, there is a single latent variable (Y^*) that is not observed and which has a conditional normal distribution:

$$Y^* = X\beta + \varepsilon \qquad\qquad \varepsilon \sim Normal\ (0,1)^{20}$$

The observed data is assumed to be generated from Y^* in the following way:

$$Y = 0 \quad if\ \mu_0 \le X\beta + \varepsilon < \mu_1$$

$$Y = 1 \quad if\ \mu_1 \le X\beta + \varepsilon < \mu_2$$

$$Y = 2 \quad if\ \mu_2 \le X\beta + \varepsilon < \mu_3$$

and so forth (*Y* is allowed to take on an arbitrary but finite number of values). The smallest μ is assumed to be negative infinity and the largest μ is assumed to be positive infinity, while the remainder are parameters to be estimated. However, without loss of generality we can set μ_1 equal to zero, so the model is

[20] The variance is normalized to one because the scale of the underlying dependent variable is not observed, just as in the binary Probit model.

normalized in the same way as the binary Probit model when *Y* takes on only two values.[21] Therefore the total number of μs to be estimated is the number of values which *Y* takes on less 2. The estimated μs (called MU) will be shown in the table of coefficient estimates in the output.

Among other uses, Ordered Probit models are useful for estimating the determinants of response to questions posed with a Likert scale. For example, suppose the responses to a survey question about the success of a joint venture are coded 1 through 5 with 1=very successful, 2=moderately successful, and so forth. Then we could define the variable ANSWER with values equal to 0 through 4 (1 through 5 would also work fine) and estimate the relationship between success and its determinants using an ORDPROB statement of the following kind:

```
ORDPROB ANSWER C X1 X2 ;
```

Interpretation of the estimated coefficients of an ordered probit model can be slightly tricky, since the effect of increasing the value of a regressor on the probability of Y taking on a particular value is that variable's coefficient multiplied by the difference in the densities at the two end points of the cell:

$$\frac{d\,\Pr(Y=j)}{dx}=[\phi(\mu_j - X\beta)-\phi(\mu_{j+1}-X\beta)]\beta$$

where φ denotes the normal distribution function. Obviously the sign of this derivative can be different from the sign of β in some cases. For further details, see Maddala (1983) or Greene (3rd edition, Chapter 19).

9.5.2 Count Data Estimation: POISSON, NEGBIN

Some dependent variables are integer counts, such as the number of patents or number of accidents. Besides being integers, they are never negative (although they may often be zero), and their variance often increases with their mean. The Poisson and Negative Binomial use an exponential mean function *exp(Xβ)* to guarantee that the mean cannot be negative. The Poisson model assumes the variance equals the mean ("equidispersion"). The Negative Binomial model

[21] This normalization implies that in general, you will wish to include an intercept (C) in your list of independent variables, just as you would for PROBIT.

assumes the variance is larger than the mean (“overdispersion”). In negative binomial model 1, the variance is proportional to the mean, while in negative binomial 2, the variance is a quadratic function of the mean.

Both of these models are similar to weighted nonlinear regressions, where the weights are functions of *exp(Xβ)*. This suggests that plain nonlinear regression could be used if the variance is independent of the mean (a form of “underdispersion”). Poisson and negative binomial estimation have the advantage over unweighted nonlinear regression in that they will fit the lower counts more tightly, at the expense of a worse fit for the higher counts. Of course, they have the additional advantage of being the most efficient estimator when the true distribution of the data is Poisson or negative binomial.

The POISSON and NEGBIN commands are invoked just like ordinary least squares, with the dependent variable followed by the list of regressors or independent variables. For example, to estimate a model where the number of patents applied for is a function of R&D spending, firm size, and the industry, use the following commands:

```
POISSON PATENTS LOGR LOGE SCISECT ;
NEGBIN (MODEL=1) PATENTS LOGR LOGE SCISECT ;
NEGBIN (MODEL=2) PATENTS LOGR LOGE SCISECT ;
```

All the usual nonlinear options may be included in these commands. The POISSON procedure produces a test for the hypothesis that the conditional mean of the data equals the conditional variance (which is an implication of the Poisson specification). If this test rejects, the coefficient estimates may still be consistent, but HCOV=N type standard errors would be quite underestimated (for this reason, POISSON uses the consistent HCOV=W by default). In this case, you may wish to consider using a negative binomial specification.

9.6 General Maximum Likelihood Estimation: ML, EQSUB

The ML procedure provides a very general estimation method. You write the likelihood function for an observation in a FRML statement, and TSP maximizes it with respect to the parameters it contains, using the data in the current sample. ML is quite powerful. It can compute analytic first and second derivatives for use in iteration and in computing standard error estimates. For example, the following commands will estimate a binary probit model (an improved version of this example is discussed later in this section):

```
FRML EQL LOGL = LOG( (1-WORK)*CNORM(-A-B*X) +
        WORK*(1-CNORM(-A-B*X)) );
PARAM A,B;
ML EQL;
```

The automatic differentiation in ML is a great advantage over coding the derivatives by hand in Fortran, Gauss, or Matlab (and especially in debugging them). The disadvantage is that execution time is slower and numerical error handling more difficult, so take care when writing the likelihood function to minimize these problems.

Execution time can be reduced by simplifying the FRML as much as possible, and eliminating repeated terms. Repeated terms are best handled with the EQSUB (equation substitution) command, which guarantees that the repeated term will be evaluated only once. EQSUB is also useful for changing the list of explanatory variables in different parts of the likelihood function, so that it is not necessary to rewrite the general equation. See the *Reference Manual* for a fuller discussion of the power of EQSUB. Numerical errors are often reduced by simplifying the function, but be careful to avoid operations like taking the log of zero.

For example, the above probit could be rewritten in two ways. The first involves minimum execution time, but may fail because of an attempt to take the log of zero. It is the model above, with the CNORM(-A-B*X) term defined in another equation, and its coefficient (called WORKYN) evaluated outside the FRML since it depends only on the data and not on the parameters. The EQSUB command is not strictly necessary here, but it is useful for changing the variables in the model and avoiding repeated evaluation.

```
WORKYN = (1-WORKING) - WORKING ;        ? recode (0,1) -> (1,-1)
FRML CNXB CNORM(-A-B*X);
FRML EQL LOGL = LOG( WORKING + CNXB*WORKYN );
EQSUB EQL CNXB;
? Substitute the expression for CNXB into the model.
PARAM A,B;
ML(HCOV=N) EQL;
```

The second way of writing the LOGL equation avoids the log of zero, but may take slightly more execution time. Generally, this approach is preferred for larger models where extreme values can result in negative arguments to the LOG function. In this example, the function LCNORM(u) =

LOG(CNORM(u)) and the relationship CNORM(u) = 1-CNORM(-u) are used.

```
NOWORK = WORKING = 0;   ? (0,1) -> (1,0)
FRML XB A+B*X;
FRML EQL LOGL = NOWORK*LCNORM(-XB) +
        WORKING*LCNORM(XB);
EQSUB EQL XB;
PARAM A,B;
ML(HCOV=N) EQL;
```

See the table below for comparative timings using these two methods. Several other considerations for writing LOGL equations for ML are discussed in the *Reference Manual*.

Figure 9.1 Comparative Timings in Seconds

Method	**Iterations**	**386**	**Pentium**	**Pentium**
		16Mhz	**266 Mhz**	**526 Mhz**
Observations		385	10,000	10,000
Canned Probit	10	5.9	2.3	0.5
ML example 1	11	98.1	9.0	3.4
ML example 2	11		10.0	4.3

As in LSQ, the likelihood function equation for the ML procedure may contain logical expressions (such as WAGE>0 or GAMMA*(1-BETA)<=0). However, be aware that if these logical expressions contain parameters to be estimated, the likelihood function will not be differentiable over the whole parameter space, and will not satisfy the usual regularity conditions that guarantee the consistency and asymptotic normality of maximum likelihood estimates. Therefore, take extra care in computing estimates with this type of likelihood function (try several sets of starting values, etc.).

9.6.1 ML PROC

Some likelihood functions are difficult to write in terms of a single FRML, especially time series models with recursive state variables, like GARCH and models with MA(q) residuals. Other models in this class are the Kalman Filter with hyperparameter estimation, simulation estimators like Multivariate Probit, and models with concentrated likelihood functions (although these can often be done in unconcentrated form). For these models, you write a PROC which evaluates the log likelihood, taking as many commands as are necessary, and store the value of the likelihood in the scalar @LOGL. You then tell ML the name of the PROC and give it a list of the parameters to be estimated. It uses numeric derivatives, so it will not be as fast as the FRML method above, but at least it is feasible. For example, the PROBIT model above can be estimated (rather slowly) with:

```
PARAM A,B;
ML MYPROB A,B;
PROC MYPROB;
        LOGL = LOG( WORKING + CNORM(-A-B*X))*WORKYN );
        MAT @LOGL = SUM(LOGL);
ENDPROC;
```

Please see the TSP web page for more advanced examples of ML PROC. There is also a simple ML PROC example in the *TSP Reference Manual* under KALMAN, which shows one way to impose inequality constraints in the Kalman Filter model.

9.7 ML examples

This section presents some examples of using ML. Some of these examples are already built into TSP, like OLSQ, so normally you will not need to use them. However, they can be convenient if you wish to modify the basic linear index function in some way, or estimate a model that has common parameters in the variance and the regression function. These examples are intended to give an idea of the power of ML, and to provide hints about how to write the likelihood function. See the previous section for an extensive Probit example and the TSP website for many more examples of how to use the ML procedure.

9.7.1 OLS

Here is ordinary least squares done with ML. This is not a recommended way to do OLSQ!

```
FRML EQ1 LOGL = LOG(SIGI) + LNORM((Y-XB)*SIGI);
FRML EQXB1 XB = B0 + B1*X;
EQSUB(NAME=OLS) EQ1 EQXB1;
PARAM B0 B1 SIGI; SET SIGI = 1;
ML(HITER=N,HCOV=NBW) OLS;
```

9.7.2 Box-Cox Transformation

Nonlinear regression is inappropriate for this model because the dependent variable is not a linear function of the disturbance. To use the ML command instead, you need to write the log likelihood, including the Jacobian.

```
LY = LOG(Y);
FRML EQB LOGL
      = LOG(SIGI) + LNORM(RESID*SIGI) + (LAM-1)*LY;
EQSUB EQB RESID;
PARAM A,1 B,1 LAM,.5 SIGI,1;
ML(HITER=N,HCOV=NBW) EQB;
```

Note that it is easier to do Box-Cox via FIML, since it will automatically include the same Jacobian term as above. The unnormalized equation for Box-Cox is the following:

```
FRML RESID (Y**LAM - 1)/LAM - (A + B*X);
FIML(ENDOG=Y) RESID;
```

9.7.3 Frontier production function model

The model is the following:

$$y = f(x,b) + e$$

$$e = -u + v$$

$$v \sim N(0,\sigma^2); \qquad u>=0$$

For a frontier cost model use $e = u + v$. See Maddala (1983) pp. 194-196. Note that the regularity conditions for asymptotic normality are not satisfied by this model.

```
FRML RESID E = Y - A - B*X;
PARAM A,B;
FRML FRONTP LOGL = LOG(2) + LOG(SIGI) + LNORM(E*SIGI) +
        LCNORM(-E*LAMBDA*SIGI);

? for a cost model, there would be no minus sign here.
PARAM LAMBDA,SIGI;
EQSUB FRONTP RESID;
```

```
? crude starting values from OLS -- there are better ones available
OLSQ(SILENT) Y C X; UNMAKE @COEF A,B;
SET SIGI = 1/@S; SET LAMBDA = .1;

ML FRONTP;
```

9.7.4 Nested Logit

This example does nested Logit using ML, and includes an example of testing for IIA. See the web site http://www.tspintl.com for a more complex and general example.

The example below is for the simplest nested logit model, where the top branch is a choice between alternative 1 (with characteristics X1) and the lower branch. The lower branch is a choice between alternative 2 (with char. X2) and alternative 3 (with char. X3). Alternatives 2 and 3 are correlated; the inclusive value parameter is denoted lambda. Note that the coefficients of the lower branch must be multiplied by lambda to obtain estimates that can be compared to ordinary MN logit.

The likelihood for an individual observation is

LOGL = D(1 chosen)L(1|X) + D(2 chosen)L(2|X)
+ D(3 chosen)L(3|X)

where D(.) is a zero-one variable denoting which choice was made, and L(i|X) is the likelihood of that choice conditional on the Xs.

Useful references are Maddala (1983), p. 71 (with lambda=1-sigma) and McFadden (1987), pp. 63-82.

```
? Define Nested Logit Model.
DOT 1-3 ;
        FRML XB.N (BETA.2+BETA1*X.1+BETA2*X.2)/LAMBDA ;
ENDDOT ;
FRML SUM23 EXP(XB2N) + EXP(XB3N) ;
FRML DENOM EXP(XB1N) + SUM23**LAMBDA ;
FRML NLOGIT -LOG(DENOM) + Y_1*XB1N
           + Y_2*((LAMBDA-1)*LOG(SUM23)+XB2N)
           + Y_3*((LAMBDA-1)*LOG(SUM23)+XB3N) ;
EQSUB NLOGIT DENOM SUM23 XB1N-XB3N ;
```

```
TITLE "Multinomial Logit using Nested Logit Likelihood' ;
PARAM BETA02 -1 BETA03 -1 BETA1 1 BETA2 1 ;
CONST BETA01 0 ;     ? One of the intercepts must be set to zero.
CONST LAMBDA 1 ; ? Collapses to MN Logit model when lambda=1.
? These estimates are for the null hypothesis.
? They are efficient if IIA holds (lambda=1).
ML NLOGIT ;
? Save results for the Hausman-McFadden test below.
COPY @LOGL LOGL0 ;
COPY @COEF BETAE ;
COPY @VCOV VCOVE ;

TITLE 'Nested Logit Model with upper branch 1, (2,3)' ;
PARAM LAMBDA ;
ML NLOGIT ;
COPY @LOGL LOGL1 ;

TITLE 'Wald-type test for IIA' ;
FRML WALD4IIA 1-LAMBDA ;
ANALYZ WALD4IIA ;

TITLE 'Likelihood ratio test for IIA' ;
SET LR4IIA = -2*(LOGL0-LOGL1) ;
CDF (CHISQ,DF=1) LR4IIA ;

? Hausman-McFadden Test for IIA (dropping first alternative).
? Note that proc haustest adjusts for coefficients that are not
? identified in the reduced (consistent) model. (beta02)
? First we estimate the lower branch of the model; estimates are
? consistent even if IIA property does not hold.
FRML NLOGIT2 -LOG(SUM23) + Y_2*XB2N + Y_3*XB3N ;
EQSUB NLOGIT2 SUM23 XB2N XB3N ;
CONST LAMBDA 1 ;
SELECT Y>1 ;                    ? Sample: subset that chooses 2 or 3.
ML (SILENT) NLOGIT2 ;

TITLE 'Hausman-McFadden Test for IIA' ;
COPY @COEF BETAC ; COPY @VCOV VCOVC ;
? See section 13.4.1 for this procedure.
```

```
HAUSTEST BETAE VCOVE BETAC VCOVC ;
```

9.7.5 Switching regression

A disequilibrium model can be estimated using switching regression. The observed quantity, Y, is the minimum of the quantity supplied (equation 1) and the quantity demanded (equation 2). The sample selection is unknown, i. e., for any given observation on price and quantity, it is not known whether it is supply or demand constrained. Reference: Maddala (1983), p.298 (10.21).

```
FRML U1EQ U1 = (Y-(B01-B11*P-B21*X1)) ;
FRML U2EQ U2 = (Y-(B02-B21*P-B22*X2)) ;
FRML LOGLEQ LOGL = LOG((NORM(U2/SIG2)/SIG2) *
        (1-CNORM(U1/SIG1))
                  + (NORM(U1/SIG1)/SIG1) *
        (1-CNORM(U2/SIG2))) ;
EQSUB LOGLEQ U1EQ U2EQ ;

? Use OLS to obtain starting values for ML estimation.
OLSQ Y C P X1 ;
UNMAKE @COEF B01 B11 B12 ;
SET SIG1 = @S ;
OLSQ Y C P X2 ;
UNMAKE @COEF B02 B21 B22 ;
SET SIG2 = @S ;

? Now do MLE on switching regression model.
PARAM SIG1 SIG2 B01 B02 B11 B12 B21 B22 ;
ML LOGLEQ ;
```

9.7.6 Bivariate probit model

```
? Uses the CNORM2(z1,z2,rho) function.
? Make the X*B1 and X*B2 equations, in the form
? FRML EQ1 XB1 = B1_0 + B1_X1*X1 + B1_X2*X2 + ...

FORM(VARPREF=B1_,PARAM) EQ1 XB1 XS;
LIST B1S @RNMSF;
? PARAM names from FORM are stored in @RNMSF
FORM(VARPREF=B2_,PARAM) EQ2 XB2 XS;
LIST B2S @RNMSF;
```

```
? Log likelihood, which uses CNORM2().
? Notation follows Greene's "Econometric Analysis", 2nd ed., p.661.
FRML BIVPROB LOGL = LOG( CNORM2(Q1*XB1, Q2*XB2,
        Q12*RHO) );
EQSUB BIVPROB EQ2 EQ1;

? The Q1 and Q2 variables exploit the symmetry of the bivariate
? normal density. They allow the probability to be evaluated by
? CNORM2, which integrates the lower left quadrant. The probability
? of all but Y1=0 & Y2=0 are other quadrants, but they can be
? translated to the lower left quadrant by using Q1 and Q2.
Q1 = 2*Y1 - 1; ? Q1=1 IF Y1=1; Q1=-1 IF Y1=0;
Q2 = 2*Y2 - 1;
Q12 = Q1*Q2;
PARAM RHO;

? Starting values via individual probits
PROBIT(SILENT,MILLS=MILLS1) Y1 XS; UNMAKE @COEF B1S;
PROBIT(SILENT,MILLS=MILLS2) Y2 XS; UNMAKE @COEF B2S;
CORR(SILENT) MILLS1 MILLS2; SET RHO = @CORR(2,1);
ML(HITER=N,HCOV=WN) BIVPROB;
```

9.7.7 Hazard function

This log-linear hazard rate example is a variation of the Weibull model. See Lancaster’s book (1997) for further information.

P(X,t) is the conditional probability that an individual who is employed at time t is still employed at time t+1 (employment hazard rate).

*P(X,t) = 1/[1+exp(Xb + a*t)]*

For a person who become unemployed at time T, the likelihood function is

*P(X,1)*P(X,2)*...*P(X,T-1)*[1-P(X,T)]*

```
SMPL 1 NOBS ;
T = INT(EXP(LOGT)+.999) ;
MSD T LOGT X ;
HIST (DISCRETE,NBINS=100,WIDTH=1) T ;
SMPLIF T>=0 & T<21 ;

? The program below shows how to automate the likelihood function
? construction if you know the maximum number of periods of
? observation.
? FRML HAZARD LOGL = TERMS +(XB + A*T) - LOG[1+EXP(XB +
        A*T)];
? TERMS + LOG[1-P(X,T)]

FRML TRICK MORE = TERMS;
SET TI = 0;
DOT 1-20;                    ? E.g., the maximum for T is 20
SET TI = TI+1;
SET T. = TI;                 ? T1=1, T2=2, etc.
FRML TERMS -(T>T.)*LOG(1+EXP(XB + A*T.)) + MORE;
? this term is non-zero if t > t.
EQSUB HAZARD TERMS TRICK;
? the trick turns more back into terms
ENDDOT;
PRINT HAZARD ;
FRML XB B0 + B1*X ;                  ? Xb formula -- user-defined
                                     ? t is the "dependent variable"
FRML LAST TERMS = 0;
EQSUB HAZARD LAST XB;

? Maximum likelihood estimation.
ML (MAXIT=50) HAZARD;
```

10. NONLINEAR METHODS AND OPTIONS

The estimation procedures described in the previous chapters -- LSQ, FIML, PROBIT, TOBIT, SAMPSEL, LOGIT, and ML, involve iterative methods. In this chapter, we describe the methods used to converge to a solution. Many options are available in the event that convergence is difficult to obtain, while other options allow the calculation of alternate standard errors for the estimated parameters.

10.1 Nonlinear methods for estimation

All of TSP's nonlinear statistical techniques involve the minimization of a criterion function, Q, over the parameters. For single equation least squares the criterion is the sum of squared residuals, while other estimators use more complicated criteria such as (minus) the log of the likelihood function. The minimization strategy is always the same and we give a brief summary of it here. In general, the methods used are gradient methods and make use of analytic derivatives of the model obtained internally by TSP. The user may request numeric derivatives with the GRAD= option (see the *Reference Manual* for details).

Consider the objective function Q as a function of the parameter vector $\boldsymbol{b}$: $Q(\boldsymbol{b})$. Let $\boldsymbol{d}$ be a vector of the same dimension as $\boldsymbol{b}$ with the property that Q decreases in the direction defined by $\boldsymbol{d}$. By a Taylor series expansion, $\boldsymbol{d}$ will be approximately equal to $\boldsymbol{H}^{-1}\boldsymbol{g}$, where $\boldsymbol{g}$ is the gradient of the function Q with respect to $\boldsymbol{b}$ and $\boldsymbol{H}$ an approximation to the Hessian matrix (the second derivatives of Q with respect to $\boldsymbol{b}$). $Q(\boldsymbol{b}+\varepsilon\boldsymbol{d})$ will be a decreasing function of the scalar ε, at least for very small values of ε, since $\boldsymbol{g}$ is a gradient and $\boldsymbol{H}$ is positive definite.

TSP proceeds iteratively in the following way: at the beginning of iteration i, parameter values $\boldsymbol{b}^{i-1}$ are available. TSP computes the change vector $\boldsymbol{d}$ from $\boldsymbol{g(b)}$ and $\boldsymbol{H(b)}$. Next it checks for convergence, defined as sufficiently small elements of $\boldsymbol{d}$. For each parameter (indexed by j), $|d_j|$ must be less than a prescribed tolerance, TOL, or $|d_j/b_j^{i-1}|$ must be less than the same tolerance. The exact formula used is

$$| d_j | \leq TOL | b_j^{i-1} | + TOL * 10)$$

If convergence has not been achieved, TSP goes on to find a better set of parameters, $\boldsymbol{b}^i$. It first tries

$$b^i = b^{i-1} + d$$

If $\boldsymbol{b}^i$ is better, i.e., if

$$Q(b^i) < Q(b^{i-1}),$$

the iteration is complete and the next one begins with the parameter $\boldsymbol{b}^i$. Otherwise, TSP begins to try parameter vectors of the form

$$b^i = b^{i-1} + \varepsilon\, d$$

first for the stepsize $\varepsilon = 1/2$, then $\varepsilon = 1/4$, and so on. This process is called "squeezing" and continues until a better $\boldsymbol{b}^i$ is found or a squeeze limit is exceeded. If squeezing is successful, TSP goes on to a new iteration. If it fails, iteration stops and a message is printed. The whole iteration process continues until convergence, or until the iteration limit, MAXIT, is reached. Results are printed whether or not convergence has been achieved, since the process is often close to convergence (how close can be seen by examination of the last value of CRIT=, which is approximately equal to the average error in the parameters squared). These results need to be interpreted with care, and are statistically meaningless if the process is far from convergence.

10.2 General convergence hints

Several options control this iteration process, and may be useful for reaching convergence in highly nonlinear or barely identified models. In general, starting values are most critical for success with difficult problems, but iteration methods may help somewhat. Starting values obtained from alternate consistent estimators, such as those based on smaller incomplete models are usually useful. An example would be to use single equation estimates before estimating all the equations jointly. Problem parameters are identified by checking the CHANGES (***d***) row in the output for especially large values. These parameters can be held fixed by use of the CONST statement, reducing the dimension of the parameter vector until the remaining parameters have converged. Then the problem parameters can be re-entered with a PARAM statement, possibly with additional experimentation in their starting values. Large CHANGES values can also indicate lack of identification in the current data set, even if the model is theoretically identified.

Grid searches can be performed to obtain starting values for critical parameters. DO loops are ideal for this purpose, since the index and step variables need not be integers (see Section 12.1 for more information on DO).

For example, AR1(METHOD=HILU) Y C X; could be done with:

```
FRML EQA Y = A + B*X + RHO*(Y(-1)-A-B*X(-1));
PARAM A,B;
DO RHO = -.95 TO .95 BY .1;
  LSQ EQA;
ENDDO;
```

Iteration options can also help obtain convergence for some models. These options are explained in detail below; they are also discussed in the NONLINEAR section of the *Reference Manual*.

10.3 Diagnostic printing: the print options

The first step in tackling a difficult convergence problem is to insure you are getting enough diagnostic output.

The SILENT option suppresses all iteration output, except a possible non-convergence message with full display of function values and changes in the last iteration. It is useful only when convergence is certain. A more useful option for general use may be the TERSE option, which reduces the printout to only the value of the criterion function at convergence and the usual table of estimated coefficients and their standard errors. This option is convenient when you are performing a series of estimations and already know that the model is well-behaved.

Slightly more output is provided by the default NOPRINT option, which displays the starting values. For each iteration, it prints the function values $F = Q(\boldsymbol{b}^{i-1})$ and $FNEW = Q(\boldsymbol{b}^{i})$, squeeze level (ISQZ, STEP or STEPSIZE = ε), and gradient norm (CRIT or CRITERION = $\boldsymbol{g'H^{-1}g}$ or $\boldsymbol{d'Hd}$). The gradient norm is the length of the gradient in the metric of the Hessian approximation. This should always decrease; otherwise, convergence will be difficult to obtain. At convergence, the number of iterations and function evaluations (the value of Q corresponding to different values of $\boldsymbol{b}$) is displayed.

The PRINT option includes all the above output, along with current parameter values and their changes (the $\boldsymbol{d}$ vector) at each iteration. The complete list of iteration options is also displayed at the start.

The VERBOSE option displays $\boldsymbol{g}$, $\boldsymbol{H}$, and $\boldsymbol{H}^{-1}$ at each iteration and is useful mainly for debugging especially difficult problems.

10.4 Numerical error handling

Some nonlinear equations involve functions sensitive to bad or unrestricted parameter values. For example, SQRT is defined only for non-negative values, LOG is defined only for positive values, and EXP typically causes problems for arguments larger than about 88 (on most computers). Again, starting values are important, but some diagnostic output is provided to help solve the problem. When numeric errors are encountered, the offending observation number and argument value are printed for the first 10 occurrences. If this occurs during a derivative evaluation, the number(s) of the parameter(s) involved are printed.

TSP requires that the starting values be good enough so that no numeric errors are encountered in the initial function and derivative evaluation, although it allows them during the stepsize search. A valid *Q* value is required for testing iterative improvement, while a valid ***d*** (i.e., ***g*** and ***H***) is required for calculating $\mathbf{b}^i$. Otherwise it is not possible to do any iterations. If there are outliers in the dataset, it may be useful to drop them from the sample (at least until better starting values are found).

During iterations, numeric errors in evaluating *Q* are not a problem, since ε is squeezed repeatedly until the numeric errors stop. Eventually $\boldsymbol{b}^{i}$ will be close to the valid $\boldsymbol{b}^{i-1}$ if squeezed far enough. Numeric errors in derivatives are still a potential problem, but are less likely to occur since the derivatives are only evaluated when a valid and improved *Q* has been found.

If a problem requires restricting a parameter to a certain range, often the best solution is to enter the parameter in a functional form, making this restriction explicit. For example, if a parameter is restricted to be non-negative, it can be entered into the equation(s) as its square (for example, B*B instead of just B). To restrict a parameter B to the range *[a,b]*, use the function *a+(b-a)*CNORM(B)*. ANALYZ can be used after estimation to obtain a standard error for the parameter of interest using the delta method.

You may find writing this parameter using EQSUB useful in this case. For example, suppose you want to estimate a regression model where the parameter B lies between zero and one. Use the following commands:

```
FRML B CNORM(D) ;       ?Constrains B; D can take on any value.
FRML EQ Y = A + B*X ;  ? Model and substitution for B.
EQSUB EQ B ;
```

```
PARAM A D ;          ? A and D are parameters of model.
LSQ EQ ;             ? Estimate
ANALYZ B ;           ? Derive estimate of B and its standard error.
```

10.5 Hessian and gradient methods: HITER, HCOV

Iteration algorithm options give variations in computing ***H*** and ***g***, the components of ***d***. (For more information on the numerical analysis of the problem see Gill, Murray, and Wright 1981.) The HITER=B or N or G or D option specifies the method of approximating ***H*** during the iterations. This may be different from HCOV=, which specifies the method of approximating ***H*** when computing the covariance matrix (and thus standard errors) for the parameters at convergence. The HCOV= option allows for printing more than one set of standard errors (and asymptotic t-statistics) for a given set of estimated coefficients, by specifying an option like HCOV=NBW instead of just HCOV=N. Depending on the estimation command (LSQ, FIML, PROBIT, ML, etc.), the available options for HITER= and HCOV= vary. Check the *Reference Manual* for further details.

HITER=B ("BHHH") specifies use of the Berndt-Hall-Hall-Hausman method. It uses the covariance of the analytic gradients for each observation to form ***H***. It has the advantage of being easy to compute and is guaranteed non-negative definite as long as the number of observations is greater than the total number of parameters.

HITER=N ("Newton") uses analytic second derivatives to form ***H***. This may be time-consuming to compute for nonlinear models, but provides faster convergence near the solution.

HITER=G ("Gauss") is standard for LSQ and FIML, and involves a quadratic form in the derivatives of the residuals with respect to the parameters, around the estimated residual covariance matrix.

HITER=D ("DFP") uses the Davidon-Fletcher-Powell method to update ***H*** at each iteration based on the gradient and parameter changes (the initial ***H*** is an identity matrix). It also implies the use of numeric derivatives to compute ***g***. Analytic derivatives are the default method for computing ***g***. HITER=D is only useful in the most extreme cases, for bad starting values, because it is very slow. The SYMMETRI option specifies use of the most accurate (and time-consuming) numeric derivatives.

HITER=F (“BFGS”) uses the Broyden-Fletcher-Goldfarb-Shanno algorithm with analytic first derivatives and a rank one update approximation to the Hessian (like DFP, but somewhat improved).

There are two HCOV= options that are not used for HITER= :

HCOV=W ("Eicker-White") computes standard errors based on a combination of the BHHH and Newton matrices. Useful for some forms of misspecification in maximum likelihood estimation and for heteroskedastic errors in all estimation procedures.

HCOV=R ("Robust") computes standard errors robust to heteroskedasticity. Used in LSQ only, this is equivalent to the old ROBUST option, and is a synonym for the W option.

10.6 Squeezing: STEP, MAXSQZ

The algorithm that generates successive values of the stepsize is chosen by the STEP option. The default STEP method depends on the HITER option and on the estimation procedure. The available methods are CEA, BARD, CEAB, BARDB, and GOLDEN.

STEP=CEA is the simplest method. ε = 1, .5, .25, ..., 2**(-ISQZ). This is the default for HITER=N and HITER=B (procedures like PROBIT and ML).

STEP=BARD uses a local quadratic approximation to the function value based on the previous value of ë. It is also bounded in the interval [.75ε,.25ε]. A typical sequence could be ε = 1, .25, .0625, ... This is the default for LSQ with HITER=G.

STEP=CEAB is the same as CEA, but if ε = 1 improves the objective function, ε = 2, 4, ... etc. are used to see if they result in further improvement. This method may be useful if ε = 1 is always improving the function, but only slowly, and the elements of d always have the same sign. This is FIML's default with HITER=G.

STEP=BARDB is the analogous extension to ε > 1 for BARD.

STEP=GOLDEN is a bracketing method that tries smaller and larger ε values even after a ε has been found which improves the objective function. The bracketing stops with the current best value of ε when MAXSQZ is reached or when ε has been determined up to the tolerance specified in the SQZTOL option. The default value of SQZTOL is 0.1. GOLDEN is the default for HITER=D.

The MAXSQZ option limits the total number of ε values attempted in a given iteration. The default value of MAXSQZ is 10 for all STEP options except GOLDEN, where the default value is 20. If ISQZ > MAXSQZ, the message "FAILURE TO IMPROVE OBJECTIVE FUNCTION (MAXSQZ)" will appear. Increasing MAXSQZ will not automatically eliminate this problem; first check the CHANGES row to see which parameters are causing the trouble. Better starting values for those parameters may help.

10.7 Iteration options: MAXIT, TOL

The MAXIT option specifies the maximum number of iterations. The default is 20 for all procedures. If MAXIT is exceeded, the message "CONVERGENCE NOT ACHIEVED AFTER n ITERATIONS" will appear. It may be useful to increase MAXIT if you have many parameters. However, if the CRIT value (section 10.3) is not decreasing smoothly, the objective function may be very non-quadratic near the current parameter values, so changing MAXIT may not help (again, better starting values may help).

The TOL option checks for convergence at the start of each iteration (section 10.1). The default value is 0.001 for all procedures. Usually there is little reason to increase TOL, except possibly to reduce the number of iterations in preliminary runs. TOL can be decreased if extremely accurate parameter values are desired.

11. ESTIMATION USING TIME SERIES DATA

The analysis of discrete time series data is central to econometrics, particularly macroeconometrics. The original name of TSP itself (**T**ime **S**eries **P**rocessor) recognizes this fact. All procedures in TSP are designed to operate on time series as well as on other kinds of data. This chapter describes the procedures in TSP specific to time series data, and gives some hints on working with such data.

You have already encountered the simplest of such procedures in AR1: the Almon and Shiller distributed lag variables, and the Durbin-Watson and Durbin (1970) test procedures for autocorrelation of the disturbances of a regression model. This chapter describes many more: identification, estimation, and simulation of a simple time series process using Box-Jenkins techniques, estimation of a vector (the VAR procedure), estimation of GARCH-M (Generalized Auto-Regressive Conditional Heteroskedasticity with a conditional Mean term) models, estimation with a Kalman Filter, and testing for unit roots and cointegration (COINT/UNIT).

We first review the basics of time series in TSP (such as operations on lags and leads), and then discuss methods for estimating a single time series process using the Box-Jenkins (ARIMA) methodology. This is followed by descriptions of two other models that are widely used in the estimation of time series models: the GARCH-M model, which allows for conditional heteroskedasticity of the disturbances, and the Kalman Filter model, which is a form of time-varying parameter model. We then discuss the use of vector autoregressions (VARs) to estimate dynamic linear relationships among several time series variables. The last subject is unit root and cointegration testing.

11.1 Techniques for time series data

Chapter 3 introduced the basic features that make time series data easy to handle in TSP: the FREQ command and lagged variables. This section reviews these concepts and gives a bit more information on using time series in TSP.

FREQ is used to specify the frequency of time series data; having done that enables you to specify observations using a convenient date format such as 75:4 for the 4th quarter of 1975 (rather than having to use the sequence number of the observation). Currently, TSP allows the use of annual, quarterly, monthly, weekly or undated data, as well as a special frequency structure called panel for time series-cross section data (see Chapter 15 for more information on this). A frequency is a characteristic permanently attached to each time series (and stored with the series in any databank), so that TSP can always check that you are using series that conform (are of the same frequency) to the current working FREQ and SMPL you have specified. You can convert series from one frequency to another with the CONVERT command (see Section 11.1.1).

It is easy to lead or lag a variable in TSP. The notation X(-1), X(-2), etc. means the observation on X is one, two, or more periods prior to the current one. In computing this, the current SMPL is ignored. That is, if the SMPL is 1960 to 1972, 1976 to 1990, the value of X(-1) in 1976 is that for 1975, *not* that for 1976. In this case, if you specify X(-1) and the observation for 1975 is missing, you will receive a warning for some procedures and an error for others.

The notation X(+1) or X(1) denotes the observation on X one period after the current one. Missing values for leads are handled in the same way as for lags. One feature of TSP you may find useful if you want to include many lags in a regression or other procedure is the ability to specify a variable list with lags. For example, the expression X(-1)-X(-16) means all the variables X(-1) X(-2) X(-3) ...X(-15) X(-16). See LIST in the *Online Help System* or *Reference Manual* for more on specifying variable lists.

11.1.1 Changing the frequency of a series: CONVERT

Sometimes data is organized as monthly observations but you wish to use quarterly data for analysis. Or perhaps some of your series are quarterly but one series is available only annually. The CONVERT procedure can change a series frequency to a new frequency. It uses one of several methods: averaging the data (the default); choosing the first, last, or middle observations; or summing the data. Each method is appropriate for different kinds of data. A flow variable such as investment would be summed to obtain the total investment in the new longer period, but a stock variable might be averaged or the first or last observation chosen depending on the importance of timing in your use of the variable.

To convert a sales series from monthly to quarterly, use the following sequence of commands:

```
FREQ Q ; SMPL 75:1 82:4 ;
```

```
CONVERT (SUM) SALES ;
```

SALES was a monthly series running from 75:1 to 82:12 ; We specified the sample as quarterly and gave the new range in terms of quarters so the CONVERT procedure would know that we wanted to convert from monthly (the frequency of SALES) to quarterly (the current frequency). Because it would be confusing (and risky) to mix frequencies in a series, CONVERT ignores the current sample range and converts the whole series no matter what you have specified as the SMPL.

If you want to preserve the old version of the series and save the new one, use this form of CONVERT:

```
CONVERT(SUM) SALESQ = SALES ;
```

Be aware, however, that you will not be able to use the old series SALES in other procedures after the new frequency has been specified. Series with different frequencies cannot be mixed in the same procedure (with the exception of CONVERT, of course).

When converting from a lower frequency, straight line interpolation can be used to compute the intervening points. Use the INTERPOL option to do this. The other convert options (SUM, etc.) can also be used.

```
FREQ M ; SMPL 75:1 82:12 ;
CONVERT(INTERPOL) SALESM = SALES ;
```

FREQ W (weekly) can be converted to quarterly, but not to monthly.

11.2 Box-Jenkins (ARIMA) models

One disadvantage of using structural econometric models for forecasting is that you need to know a great deal about the variables being modeled. In particular, to obtain a simulation over several periods in the future, you need to know the values for all the exogenous variables in the model over the forecast period. For this and other reasons, some forecasters use the ARIMA (AutoRegressive Integrated Moving Average) or Box-Jenkins forecasting method. This method only uses a series' own lagged values to forecast its future values. If the series follows a stationary stochastic process without too much drift or noise, this method can work well, at least over the short term. (See Section 11.6 for discussion of testing for unit roots in the process.) ARIMA models can be thought of as a sophisticated extrapolation method.

In this section, we discuss how TSP obtains univariate ARIMA forecasts. We strongly recommend that users who are interested in this technique refer to one of the standard references for further details. The basic reference on the subject is Box and Jenkins (1976). Two other elementary books that focus on this method are Nelson (1973) and Van Daele (1983). The Pindyck and Rubinfeld text is also recommended; Section 3 of this bood is entirely devoted to time series models.

Box-Jenkins forecasting is traditionally divided into three parts: identification (determining the form of the time series process the variable follows), estimation (estimating the parameters of the process), and forecasting (extrapolating the process beyond the estimation period using the estimated parameters). In TSP these three steps correspond to the procedures BJIDENT, BJEST, and BJFRCST. Because these procedures have most of the same options, and share their values, once you have specified a set of options for BJIDENT, they are automatically assumed for any subsequent Box-Jenkins procedure, unless explicitly changed.

11.2.1 Identification: BJIDENT

Suppose you have a product's monthly sales data from 1978 through the middle of 1987, and you assume that whatever time series process generated the data is stationary (at least after differencing). Such a series is shown in **Figure 11.1**. To look at some characteristics of this process, you can use a BJIDENT command to display the autocorrelations and partial autocorrelations of the series:

```
BJIDENT (NDIFF=1,NSDIFF=1,NLAG=12,NLAGP=12) MSALES ;
```

Since you suspect that the original series may not be stationary, the NDIFF option specifies that the correlations be computed for the first differenced series, as well as for the original series. Similarly, the NSDIFF option specifies that seasonal differencing is to be performed. BJIDENT determines the periodicity of the seasonal factor from the frequency of your current sample; in this case the FREQ is monthly, and observations 12 periods apart would be differenced. If you have not specified a FREQ as part of your SMPL, you will have to supply the seasonal span as the NSPAN option on the BJIDENT command.

NLAG and NLAGP options specify the number of autocorrelations and partial autocorrelations to be computed, that is, the length of the lag over which they are to be computed. The default value is 20.

BJIDENT's output consists of plots of the series, both differenced and undifferenced, followed by a printout of autocorrelations and partial autocorrelations, their standard errors, and Q-statistics (Ljung-Box statistics, see Harvey (1993), p. 212) for the hypothesis that all autocorrelations of higher order are zero. Correlations are also plotted in what is called a correlogram and a partial correlogram. These can be quite useful in trying to determine the form of the process. Consult Box and Jenkins or Nelson for examples of the correlograms associated with various time series processes. The autocorrelations, partial autocorrelations, and inverse autocorrelations are also stored under the names @AC, @PAC, and @IAC.

Some of the output of BJIDENT is given in **Figure 11.2**, which shows the autocorrelations and partial autocorrelations for the series after it has been first differenced and seasonally differenced at lag 12. In our example, the series appears to be nonstationary in levels but stationary in first differences. There appears to be a seasonal effect at 12 month intervals. The correlogram for the first differenced series suggested that the data be fit by a first order moving average error process, since there were no significant autocorrelations after the first.

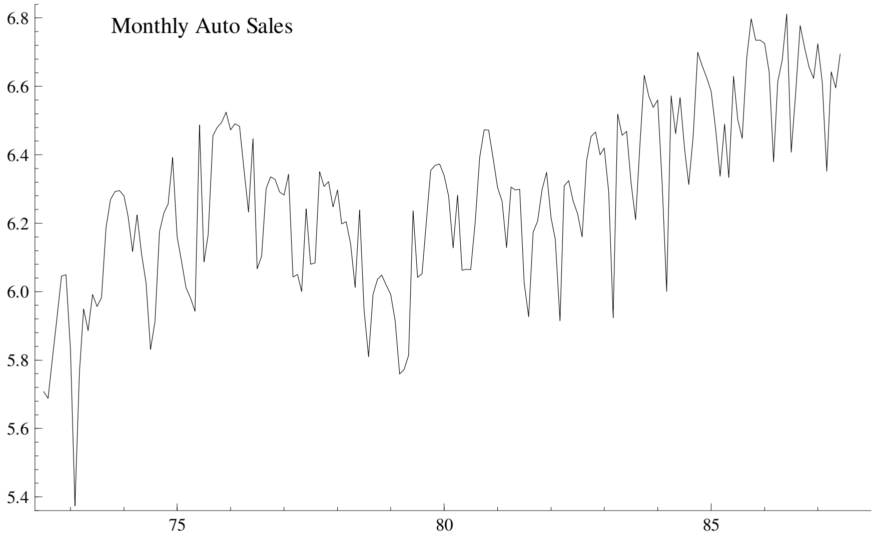
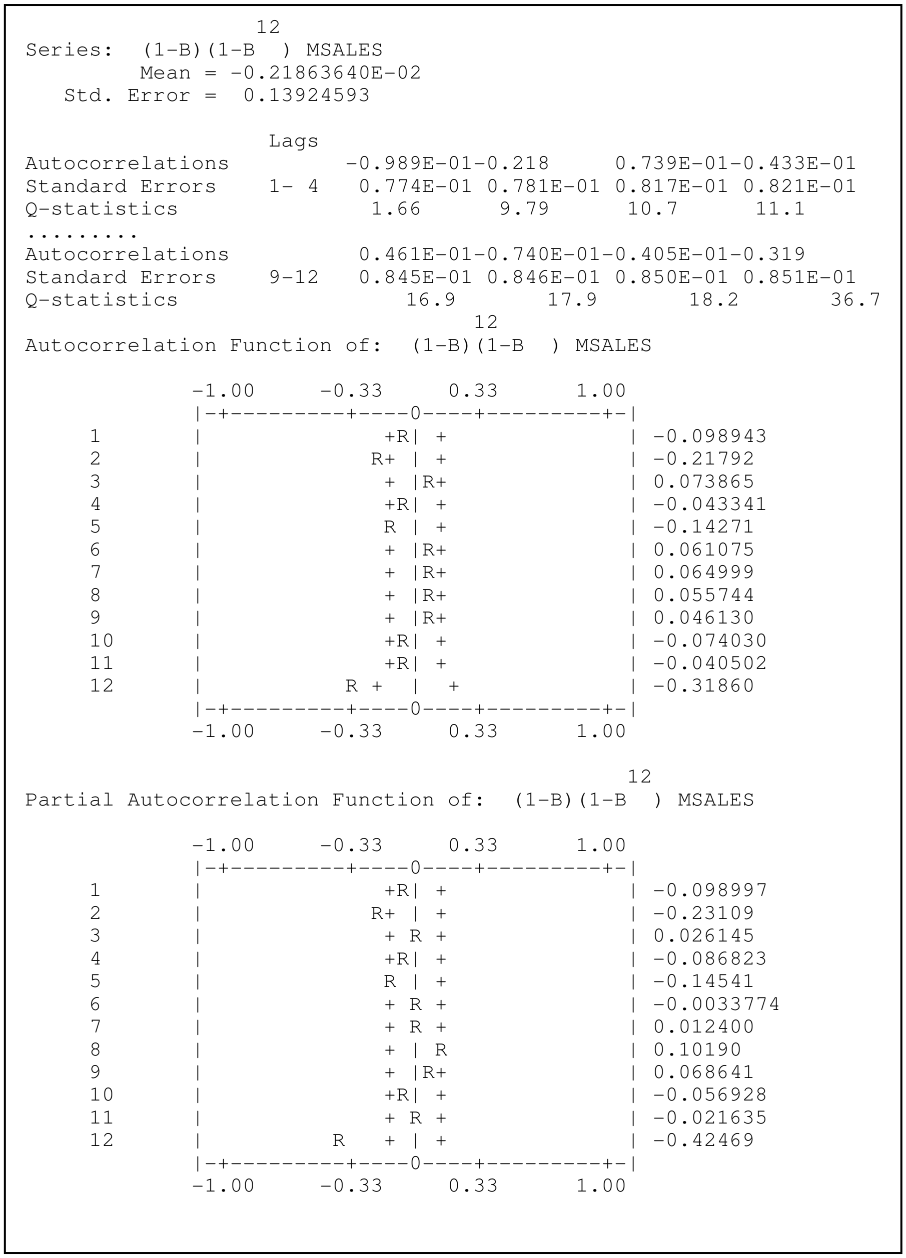


Figure 11.1 Series to be analyzed by Box-Jenkins



```
                  12
Series:  (1-B)(1-B  ) MSALES
         Mean = -0.21863640E-02
   Std. Error =  0.13924593

                   Lags
Autocorrelations         -0.989E-01-0.218     0.739E-01-0.433E-01
Standard Errors    1- 4   0.774E-01 0.781E-01 0.817E-01 0.821E-01
Q-statistics               1.66      9.79      10.7      11.1
.........
Autocorrelations          0.461E-01-0.740E-01-0.405E-01-0.319
Standard Errors    9-12   0.845E-01 0.846E-01 0.850E-01 0.851E-01
Q-statistics                  16.9       17.9       18.2       36.7
                                   12
Autocorrelation Function of:  (1-B)(1-B  ) MSALES

             -1.00     -0.33     0.33      1.00
             |-+---------+----0----+---------+-|
     1       |              +R| +              | -0.098943
     2       |             R+ | +              | -0.21792
     3       |              + |R+              | 0.073865
     4       |              +R| +              | -0.043341
     5       |              R | +              | -0.14271
     6       |              + |R+              | 0.061075
     7       |              + |R+              | 0.064999
     8       |              + |R+              | 0.055744
     9       |              + |R+              | 0.046130
     10      |              +R| +              | -0.074030
     11      |              +R| +              | -0.040502
     12      |           R +  |  +             | -0.31860
             |-+---------+----0----+---------+-|
             -1.00     -0.33     0.33      1.00

                                               12
Partial Autocorrelation Function of:  (1-B)(1-B  ) MSALES

             -1.00     -0.33     0.33      1.00
             |-+---------+----0----+---------+-|
     1       |              +R| +              | -0.098997
     2       |             R+ | +              | -0.23109
     3       |              + R +              | 0.026145
     4       |              +R| +              | -0.086823
     5       |              R | +              | -0.14541
     6       |              + R +              | -0.0033774
     7       |              + R +              | 0.012400
     8       |              + | R              | 0.10190
     9       |              + |R+              | 0.068641
     10      |              +R| +              | -0.056928
     11      |              + R +              | -0.021635
     12      |          R   + | +              | -0.42469
             |-+---------+----0----+---------+-|
             -1.00     -0.33     0.33      1.00
```

Figure 11.2 Sample output from BJIDENT

11.2.2 Estimation: BJEST

Using the results of BJIDENT as a guide to the specification of the model, you can then estimate the parameters of this model with BJEST. For our example, the appropriate command might be

```
BJEST (CUMPLOT,NBACK=5,NMA=1,NSDIFF=1,NDIFF=1) MSALES ;
```

This specifies a first-order moving average model on first differenced data, with no seasonal components. Note that you do not need to specify any options that remain the same from a previous BJ command; in this case we did not need to set NDIFF or NSDIFF to 1 if this command had followed the BJIDENT command given above.

The NBACK option specifies the number of backforecasted residuals that should be used to start the process and generate the initial conditions; since we are using a moving average process of order one here, only the first will be nonzero, so NBACK can be set to a small number. In the case of autoregressive or mixed processes, you should use a much larger value. Other options control the behavior of the iteration process and the appearance of the output; consult the *Reference Manual* to learn more about them.

In BJEST's output, the usual TSP estimation results are shown for the residuals from the fitted model, along with a table of parameter estimates (only two in this case, the moving average parameter θ and the seasonal moving average parameter δ), followed by some statistics concerning the variable in question). In the example, the model that fit the data was the following:

$$(1-L)(1-L^{12})S_t = (1-\underset{(.072)}{.365}\,L)(1-\underset{(.055)}{.625}\,L^{12})\varepsilon_t$$

where ε is a white noise process and L is the lag operator ($Ly_t = y_{t-1}$).[22] This is followed by a plot of the cumulated periodogram of the residuals so you can see how well the white noise assumption of the model is satisfied. In our example, both the periodogram and the Q-statistics for serial correlation suggest that the residuals are not quite white noise, although they are close.

11.2.3 Forecasting: BJFRCST

Immediately following a Box-Jenkins estimation, you can perform a forecast based on the estimated time series model using the BJFRCST command. This

[22] Box and Jenkins use the notation B for the backward lag operator and we follow them in the TSP output. This manual uses lag notation that is more familiar to economists, L.

forecast will be computed starting in any time period or range of time periods you specify; it will go forward for the number of periods given by the NHORIZ option. Confidence bounds will also be computed for the forecast. An example for the series we modeled above is:

```
BJFRCST (NHORIZ=10,ORGBEG=87:6,ORGEND=87:7) MSALES ;
```

This computes two 10-month forecasts, starting in June and July of l987. Unless requested not to, BJFRCST will also plot the forecasts and their standard error bounds.

The plot of output from this BJFRCST is shown on the following pages in **Figure 11.3.** The model chosen for the variable MSALES was the one estimated in the previous section. This model is used to generate the forecasts, using the actual value in June l987 as an initial condition for MSALES(-1). Note the 95% confidence bounds that are also plotted. As may be expected, the information contributed by this model, roughly a random walk with a large measurement error, is very small, and the standard errors are correspondingly large.

If you do a forecast based on some previously estimated time series process, you can also supply the parameters of the model directly to BJFRCST. Consult the *Reference Manual* or online *Help System* for further details on the notation (which is that of Box and Jenkins). An example for a simple moving average process is:

```
BJFRCST (NMA=1, NSDIFF=0, NDIFF=1, NHORIZ=12, ORGBEG =
        84:1, ORGEND = 84:3) SALES  S 4.021  THETA(1) -0.78 ;
```

In this command, the first difference of SALES will be forecasted for the period January 1984 to March 1984 using an MA(1) model with a coefficient of -0.78 and a disturbance whose standard error is 4.021:

$$Y_t = Y_{t-1} + e_t - 0.78e_{t-1} \qquad e_t \sim \mathrm{N}(0, 4.021^2)$$

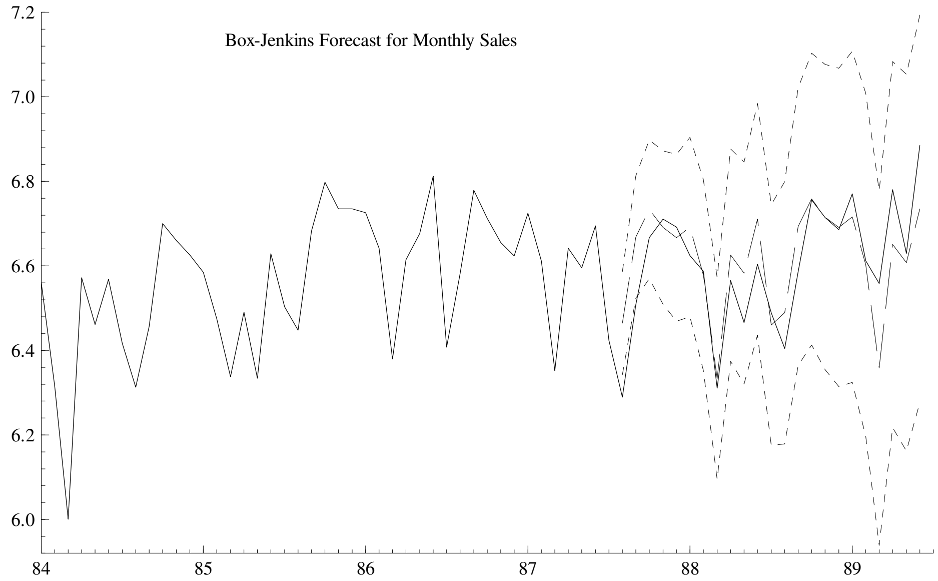


Figure 11.3 Sample Output from BJFRCST

11.3 *Auto-Regressive Conditional Heteroskedasticity*

The ARCH model originated by Engle, and its many elaborations, are widely used in econometrics to estimate models of time series processes where the variance of the disturbance is dependent in a simple way on the behavior of the preceding observations, but the conditional mean of the disturbance is still equal to zero. ARCH processes appear to describe many observed macroeconomic data series, such as exchange rates and stock market returns, quite well.

ARCH will estimate not only the basic ARCH model, but also more complicated models which allow the variance of the disturbance to follow both an autoregressive and a moving average process and to be conditional on other series. This latter feature makes ARCH suitable for heteroskedastic data of any kind, not just time series. In addition, the conditional mean of the dependent variable can depend on the standard deviation of the variance of the disturbance (the full GARCH-M model). These models are more fully discussed in Engle (1982, for ARCH), Bollerslev (1986, for GARCH and its identification), and McCurdy and Morgan (1988, for GARCH-M). A fuller discussion of models TSP can estimate is given in the *Reference Manual* (under the ARCH command).

Here is an example of simple ARCH model estimation of an index of stock prices, where there are no independent variables, and the variance of the disturbance is assumed to follow an autoregressive process of order 3:

```
ARCH(NAR=3) RM C ;
```

This command estimates the following model by maximum likelihood (denoting the stock price series RM by y_t):

$$y_t = \eta_0 + \varepsilon_t$$

$$\varepsilon_t \sim N(0,h_t)$$

$$h_t = a_0 + a_1 \varepsilon_{-1}^2 + a_2 \varepsilon_{t-2}^2 + a_3 \varepsilon_{t-3}^2$$

The coefficients estimated by this command are η_0, the average stock return, and $\alpha_0, \alpha_1, \alpha_2, \alpha_3$, the parameters of the ARCH process. These coefficients are labeled ALPHA0, ALPHA1, etc. in the output. Because the estimation of ARCH models can be prone to numerical problems arising when estimated variances are negative, zero, or very large, the ARCH procedure constrains ALPHA0 to be nonnegative, and the other ALPHAs to be bounded between zero and one. See the *Reference Manual* for details on the methods used to impose these constraints.

ARCH models can also have regressors; that is, the conditional mean of the dependent variable can be a function of other variables, just as in ordinary linear regression. For example, consider a market beta model where the return on a stock (R) is a function of the return on the market (RM), and the variance of its return follows an ARCH process:

```
ARCH(NAR=3) R C RM ;
```

Because of ARCH's ability to estimate a model where the variance of the disturbance depends on a set of regressor variables, ARCH can also be used to estimate an ordinary weighted least squares model by ML, even when there is no dependence across the observations. For example, here is a model with size-related heteroskedasticity:

$$
\begin{aligned}
y_t &= \eta_0 + \eta_1 x_{1t} + \eta_2 x_{2t} + \varepsilon_t \\
\varepsilon_t &\sim N(0, h_t) \\
h_t &= a_0 + a_1 g_t
\end{aligned}
$$

where g_t is the size of the t^{th} observation.

ARCH can estimate this model (called OLS-W by TSP) with the command

```
ARCH (GT=G) Y C X1 X2 ;
```

G denotes the series that contains the size variable. The GT option is used to specify a list of series that will enter into the regression function for the variance h_t. Note that the constant (C) is automatically included in this list of series via the a_0 parameter, so you should not include it.

11.4 The Kalman Filter (KALMAN)

Strictly speaking, the term Kalman Filter refers to an estimation method commonly used to estimate "state-space" models, rather than the model itself. This class of models consists of two parts: the transition equation, which describes the evolution of a set of state variables, and the measurement equation, which describes how the data actually observed is generated from the state variables. Its importance in economics is partly due to its ability to model time-varying parameters in an intuitively appealing way.

In addition, the Kalman Filter estimation method is an updating method that bases the regression estimates for each time period on last period's estimates plus the data for the current time period; that is, it bases estimates only on data up to and including the current period. This makes it useful for investigating structural change in parameters or constructing forecasts based only on historical data.

As with many time series methods economists use (such as ARIMA models), State Space Models originated in the engineering literature (Kalman 1960) and were imported into economics by Rosenberg (1968) among others. A good reference is Harvey (1981); there is also a special issue of the *Annals of Economic and Social Measurement* (October 1973) on time-varying parameters. A more elementary reference is Maddala (1977), Chapter 17.

Using the notation in Harvey (1981), the model KALMAN estimates can be written in the following way (assuming a single dependent variable for simplicity):

$$\begin{aligned} y_t &= Z_t a_t + \xi_t \\ a_t &= T a_{t-1} + \eta_t \\ \xi_t &\sim N(0, \sigma^2) \\ \eta_t &\sim N(0, \sigma^2 Q_t) \\ a_0 &\sim N(\alpha_0, \sigma^2 P_0) \end{aligned}$$

y_t is the dependent variable and there are m independent variables Z_t. The first equation is an ordinary regression equation with time-varying parameters. The second equation defines the evolution of these parameters. Note that matrices S_t and R_t in Harvey's notation have been set to identity matrices and matrix T_t to a constant matrix in this use of the Kalman Filter.

When estimating this model in TSP, you can optionally supply the matrices T (the transition matrix BTRANS), Q_t (the variance of the transition equation VTRANS), P_0 (the variance of the prior distribution on the parameter vector VBPRIOR), and the vector α_0 (the mean of the prior on the coefficients BPRIOR). If you fail to supply any of them, reasonable defaults will be used.

For example, the simplest KALMAN command looks like this:

```
KALMAN CONS C GNP ;
```

The above command estimates the regression of consumption on GNP in a recursive manner, allowing the coefficient of GNP and the constant to evolve as random walks (with a signal-to-noise ratio of 1). These coefficients are the model's "state" variables, and consumption is the measured variable. The intercept and GNP series are treated as known fixed constants. In Harvey's notation, the model is

$$\begin{aligned} y_t &= Z_t a_t + \xi_t \\ a_t &= a_{t-1} + \eta_t \\ \eta_t &\sim N(0, \sigma^2 I) \end{aligned}$$

α_0 will be estimated from the first m data observations, where m is the number of coefficients in α_t.

A slight generalization of this random walk parameter model is the Cooley-Prescott (1973) adaptive regression model, which has had some success in forecasting. You can estimate this model with the command

```
KALMAN(VTRANS=SIGMAV) CONS C GNP ;
```

SIGMAV corresponds to Cooley and Prescott's Σ_v, the variance of the errors in the transition equation. In the absence of specific prior information on the form of this matrix, they suggest using a diagonal matrix whose elements represent the relative variability of the regression's different coefficients. This idea could also be used to mix time-varying parameters with constant parameters (by setting their variances to zero) in the same regression.

KALMAN always calculates recursive residuals; these residuals can also be obtained with OLSQ when the REGOPT(CALC) RECRES; command is in effect. However, KALMAN can also print the evolving coefficients of the recursive model:

```
KALMAN(NOETRANS,PRINT) CONS C GNP ;
```

The *Reference Manual* discusses the KALMAN procedure further; useful features of the KALMAN command not discussed here but described in that manual include the stochastically convergent parameter model, the use of priors in estimating Kalman Filters, and the ability to estimate several measurement equations simultaneously.

11.5 Vector Autoregressions (VAR)

A vector autoregression model is the unconstrained reduced form of a dynamic simultaneous equations model; that is, it expresses a vector of endogenous variables as linear functions of their own and each other's lagged values. Contemporaneous and lagged exogenous variables may also be included in the system. This style of simultaneous equation modeling was introduced into econometrics by Sims (1980) and is now widely used for small to medium-sized macroeconometric models, particularly for forecasting.

Estimation of an unconstrained vector autoregression is quite straightforward, even in the presence of contemporaneous correlation of the disturbances. Consider the following VAR:

$$Y_t = B_1 Y_{t-1} + B_2 Y_{t-2} + ... + GX_t + E_t$$

where Y_t is an *n* by *1* vector of endogenous variables, the *B*'s are *n* by *n* matrices of coefficients, *G* is an *n* by *m* matrix of coefficients, X_t is an *m* by *1* vector of exogenous variables, and E_t is an *n* by *1* vector of disturbances. Because the same list of right-hand side variables (all the lags of the Y_t) appear in the *n* equations, this set of equations can be estimated consistently and *efficiently* by ordinary least squares; there is no need for joint estimation.

Although you could use OLSQ to estimate a VAR model one equation at a time (listing all the relevant lagged variables for each estimation), TSP provides the VAR command to make this process easier. VAR also provides some regression output specific to the VAR methodology, such as impulse response functions and forecast error variance decompositions. For example, suppose you wish to estimate the vector autoregression for the six variables money, real GNP, unemployment, wages, price level, and import prices described in the Sims paper. You would use:

```
VAR(NLAGS=4) M,RGNP,U,W,P,PM ;
```

The above command produces as output the regression coefficients for the six ordinary least squares regressions of each dependent variable on four lags each of M, RGNP, U, W, P, and PM (24 coefficients for each equation), along with the log of the likelihood function under the multivariate normal assumption on the disturbances. In addition, the program will display (and store under the name @IMPRES) the response of each endogenous variable to Choleski-factored shocks in the given order over ten periods (the impulse response function).

Exogenous variables may be included in the vector autoregression, and calculation of the impulse response function can be modified with options. For example, the command

```
VAR(NLAGS=2,NHORIZ=5,SHOCK=UNIT) M GNP | C CONS P ;
```

specifies that M and GNP are regressed on the lagged variables M(-1), M(-2), GNP(-1), GNP(-2), the constant, CONS, and P. The impulse response function will be calculated for only five time periods and a unit shock to the endogenous variables will be used. Except for computation of the impulse response, the VAR command above is equivalent to:

```
OLSQ M M(-1) M(-2) GNP(-1) GNP(-2) C CONS P ;
OLSQ GNP M(-1) M(-2) GNP(-1) GNP(-2) C CONS P ;
```

VAR automatically computes block exogeneity tests to see if lags of the other endogenous variables enter a given equation significantly. In the bivariate example above, these are Granger causality tests (F tests) for GNP(-1) and GNP(-2) in the M equation, and for M(-1) and M(-2) in the GNP equation.

A common use of VARs is as a base case for more restricted structural models of lag relationships of a set of endogenous variables. With TSP's ANALYZ procedure it is possible to calculate an asymptotic test statistic for a joint hypothesis on the reduced form VAR coefficients. This hypothesis can be linear or nonlinear. For example, suppose you wish to test that the coefficients on the lagged Ms and GNPs are proportional in the two VAR equations given above. You could use the following set of commands to perform the task:

```
VAR(NLAGS=2) M GNP | C CONS P ;
? Give the estimated coefficients names in a list.
LIST CNAMES B11-B14 G11-G13 B21-B24 G21-G23 ;
FRML PREQ1 B11/B21 - B12/B22 ;    ? Proportionality constraints
FRML PREQ2 B12/B22 - B13/B23 ;
FRML PREQ3 B13/B23 - B14/B24 ;
? Compute a joint hypothesis test with 3 degrees of freedom:
ANALYZ (NAMES=CNAMES,COEF=@COEF) PREQ1-PREQ3 ;
```

11.6 Testing for Unit Roots and Cointegration

11.6.1 Unit roots: UNIT

Simply stated, the test for a unit root in a time series y_t is the test that a regression of y_t on y_{t-1} yields a coefficient of one. This test is complicated by several features arising from the nonstationarity of y_t under the null hypothesis:

1) The ordinary t-statistic does not have the usual distribution, so you cannot use tables of t-statistics to find its p-value.
2) The correct distribution depends on nuisance parameters in the regression, in particular, whether the constant or the time trend is included.

In a well-known paper, Dickey and Fuller (1979) suggest a method for computing a test for a unit root in a time series, and present critical values for their proposed tests with and without a trend variable included. The method consists of running the stationary regression y_t-y_{t-1} on y_{t-1} either with or without a constant and time trend and testing whether the coefficient of y_{t-1} is zero. They provide the appropriate critical values for such a test in a table. Since the distribution of the resulting t-statistic generally depends on the value of the intercept in the model unless a time trend is included, most researchers choose to include both a constant and a time trend and then use the tables appropriate for that case.

Since then, a large literature on unit root tests has appeared, describing many alternative tests (some of which are variations of the above). The COINT command in TSP can compute 3 different types of unit root tests: the Dickey-Fuller (tau), Phillips-Perron (Z, "nonparametric"), and Weighted Symmetric. Each allows for various exogenous variables like time trends and seasonal dummies/trends, and each allows for a series of "augmenting" lags to control for additional serial correlation. See the references and the *Reference Manual* under COINT for further information.

For example, suppose you want to test whether consumption in Chapter 3's Illustrative Model has a unit root (is integrated of order 1). The following set of statements will perform the (augmented) Weighted Symmetric and Dickey-Fuller tests and print their approximate (asymptotic) P-values:

```
SMPL 49 75 ;
UNIT CONS;
```

In this example, UNIT chooses 2 augmenting lags for the WS test and 10 lags for the DF; the P-values are 0.89 and 0.73 respectively, so the null hypothesis of a unit root would not be rejected at the 0.05 level. The output of the UNIT tests for both CONS and for GNP are shown in **Figure 11.4**.

Alternately, the Dickey-Fuller test could be computed "by hand"!

```
SMPL 50 75 ;
DCONS = CONS-CONS(-1) ;
OLSQ DCONS CONS(-1) C TIME;
CDF(DICKEYF) @T(1) ;
```

The above example assumes that there is no further serial correlation since it does not add lagged y differences to the model. The resulting statistic was -1.36 with a corresponding asymptotic P-value of 0.84, so the null of a unit root is accepted at the 5 per cent level. Options for CDF allow you to compute the P-values without assuming the presence of a trend or constant. See the references and the *Reference Manual* for further details. Also be aware that the residuals from the Dickey-Fuller regression should be serially uncorrelated for the test to be valid, although they do not generally need to be homoskedastic (Phillips 1987).

The Weighted Symmetric test is recommended over the Dickey-Fuller test, because it has (sometimes only slightly) higher power (see Pantula, Gonzalez-Farias, and Fuller 1994). That is, the WS test is more likely to reject the unit root (null hypothesis) when it is in fact false. The Phillips-Perron test is a variant of the Dickey-Fuller which tackles the problem of additional serial correlation in the residuals by using a GMM-type method to compute a residual variance that is "robust" to autocorrelation. The default is to compute the Weighted-Symmetric and Dickey-Fuller tests only.

11.6.2 Cointegration: COINT

Cointegration modeling of time series is a methodology pioneered by Engle and Granger (1987). Two or more time series are said to be *cointegrated* if a linear combination of them is I(0) (is stationary, or has all roots outside the unit circle) even though individually they are each I(1). Thus the hypothesis of cointegration consists of two parts: tests for I(1) of the individual series and I(0) of a linear combination. Usually the term cointegration testing refers only to the second part of the hypothesis; the test is performed *conditional* on the fact that each component series is I(1). Although this hypothesis sounds different from the hypothesis of a unit root, the practice of testing for cointegration is quite similar. TSP gives the P-values for the Engle-Granger versions of these tests in the CDF procedure under the DICKEYF option.

As an example, consider testing that real consumption and real GNP from the Illustrative Model are cointegrated. It is easy to establish that each is I(1) separately (with asymptotic P-values of .89 and .99). The TSP commands to evaluate the second part of the hypothesis are the following:

```
SMPL 49 75 ;
COINT(TERSE,ALLORD) CONS GNP;
? Note that this command performs all the individual unit root tests
        before testing for cointegration.
```

The output from this command is shown in **Figure 11.4**. When CONS is the dependent variable of the cointegrating regression, COINT chooses 9 augmenting lags, and obtains a test statistic of -2.07, which has a P-value of 0.74. When GNP is the dependent variable, 10 augmenting lags are chosen, and the test statistic and P-value are -1.17 and 0.97 respectively. So the null hypothesis of a unit root in the cointegrating regression cannot be rejected at the .05 level in either test. We can conclude that the linear combination of CONS and GNP is not I(0), so they are not cointegrated (at this significance level).

```
                            Summary of Unit root tests
                                 Test Statistics

                     GNP          CONS
Wtd.Sym.       -1.50690      -0.52491
Dickey-F       -1.75488       0.35523

                                    P-values

                     GNP          CONS
Wtd.Sym.        0.88954       0.99345
Dickey-F        0.72610       0.99648

                                 Number of lags

                     GNP          CONS
Wtd.Sym.        2.00000       2.00000
Dickey-F       10.00000      10.00000


        Summary of Engle-Granger (tau) cointegration tests

Dep.Var.       TestStat       P-value      Num.lags
GNP            -2.07152       0.74083       9.00000
CONS           -1.17088       0.96513      10.00000
```

Figure 11.4 Output for unit root and cointegration tests

If the COINT test of CONS on GNP were done manually, without the search for augmenting lags, it would look like this:

```
SMPL 49 75 ;
OLSQ CONS GNP C TIME ;     ? the cointegrating regression
SMPL 50 75 ;
DRES = @RES-@RES(-1) ;
OLSQ DRES @RES(-1) ;        ? Engle-Granger test is a
? Dickey-Fuller test on residuals from the cointegrating regression.
CDF(DICKEYF,NVAR=2) @T ;
```

In the above example, consumption is regressed on a constant, time, and GNP to obtain the cointegrating vector, residuals are constructed, and then the first-differenced residuals are regressed on the lagged residual. Under the hypothesis of stationarity, the coefficient on this variable should not be zero; the t-statistic for this hypothesis is the Engle-Granger statistic. One complication is that the actual value of the Engle-Granger statistic (although not its distribution) will be affected by the choice of left-hand variable in the first regression (consumption or GNP); COINT with the ALLORD option computes both tests.

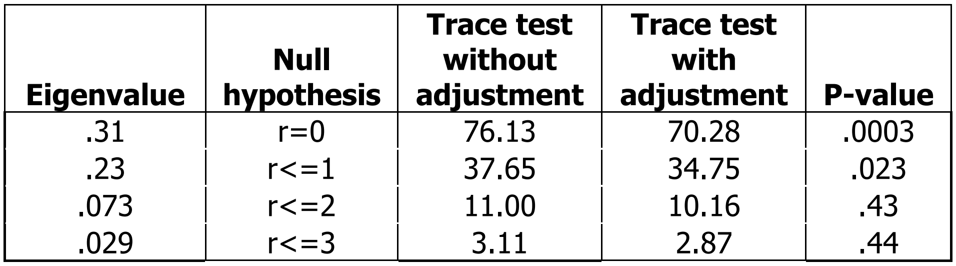
To compute the asymptotic P-value manually for the Engle-Granger statistic, use the DICKEYF option of the CDF procedure with the NVAR option to specify the number of cointegrating variables used in computing the test statistic. TSP provides P-values for cointegrating regressions with up to 6 variables, using the response surface estimates given by MacKinnon (1990, 1994). For this example, the value of the statistic with no augmenting lags was -1.364, with a P-value equal to 0.839. If we had put GNP on the left-hand side of the original regression, the corresponding value of the statistic would have been -1.366, with a P-value of 0.838, so in this case, the choice of left hand variable did not matter. The null hypothesis of a unit root in the cointegrating regression would not be rejected at the 5 per cent level, so we could not conclude that these variables are cointegrated. Note that the test can be sensitive to the estimation sample. If a period of 1950-1975 is used for the cointegrating regression rather than 1949-1975, the unaugmented P-values are .00097 and .029, which would lead to the conclusion that cointegration does exist.

COINT computes the Engle-Granger test by default. A second type of cointegration test that can be performed by the COINT command is the Johansen-Juselius (maximum likelihood) test. This involves testing for a particular restriction on the coefficient matrix of lagged dependent variables in a VAR. It is actually estimated by running two VARs and obtaining the eigenvalues for a function of their joint residual covariance matrix. Then likelihood ratio tests (with finite sample corrections) are made to check for the number of cointegrating vectors of the original system. The Johansen-Juselius test is often "oversized" (i.e. a P-value of .01 may be printed when the true rejection frequency would be about 0.10), implying that "too much" cointegration (or too many cointegrating vectors) tend to be found. For example:

```
SMPL 58:2 84:3;
COINT(JOH,SEAS,NOTREND,NOEG,NOUNIT,MAXLAG=1,MINLAG=1)
      Y1-Y4;
```

This command computes the trace tests for the Finnish data from the Johansen-Juselius(1990) paper. Note that COINT computes the tests with the finite sample adjustment, but this doesn't affect the results much (at least in this case of autoregressive order 2). In this case, we would conclude at the 5 per cent level that there are 2 cointegrating vectors (because the first 2 nulls are rejected and the third is accepted). The P-values are interpolated from Osterwald-Lenum (1992) Table 1.1* (because a constant term is included, and Table 1.1* is more conservative that Table 1). The table is shown below.

Figure 11.5 Trace Tests for Finnish Data



Eigenvalue	Null hypothesis	Trace test without adjustment	Trace test with adjustment	P-value
.31	r=0	76.13	70.28	.0003
.23	r<=1	37.65	34.75	.023
.073	r<=2	11.00	10.16	.43
.029	r<=3	3.11	2.87	.44

12. CONTROLLING THE EXECUTION OF A TSP PROGRAM

This chapter describes TSP statements that control the program's order of execution. They help advanced users who use TSP as a programming language for solving complex and nonstandard problems. Be aware that although many of the statements resemble Fortran or other high-level language statements, they will not execute as quickly as the corresponding Fortran statements would after being compiled. The advantage of programming within TSP is convenience, but this convenience comes at some cost in execution time.

The statements covered in this chapter allow you to control the amount of printed regression output (REGOPT), or the order of execution of the program (GO TO, IF, THEN, ELSE). We describe two kinds of loops: ordinary DO loops and the more flexible DOT loops. Finally, we discuss how you can write a TSP procedure that can be used in multiple TSP programs to perform a task that you do often.[23]

12.1 Loops: DO

The DO statement offers a convenient method to execute a group of TSP statements several times, using a variety of parameter values or making some other change each time. All the statements between DO and ENDDO are executed repeatedly according to information provided in the DO statement. In its fullest form, DO uses a counter variable set equal to a lower limit, increments by a fixed amount each time through the loop and stops when an upper limit is reached. The statement has the form DO followed by the name of the counter variable, an = sign, the lower limit, TO, the upper limit, BY, the increment. For example,

```
DO I=1 TO 7 BY 1;
```

specifies that I is the counter, it starts at one and goes through 7 in steps of one. Shorter forms of the DO statement are available; if BY is omitted, the increment is taken to be one; for example,

[23] If you are using TSP interactively, the commands described in this chapter (except REGOPT) must be entered in COLLECT mode, so they can be executed together. See Chapter 17 for more information on interactive use.

```
DO I=1 TO 7;
```

or

```
DO I=1,7;
```

have the same effect as the previous example. If everything except DO is omitted, the loop is executed just once. This last type of DO loop provides a useful extension of the IF statement (see section 12.5). For example,

```
OLSQ Y C,X;
IF @DW <= 1.5; THEN; DO;
        SET RHO = 1.0-@DW/2 ;
        GENR YT = Y - RHO*Y(-1);
        GENR XT = X - RHO*X(-1);
        GENR CT = 1-RHO;
        OLSQ YT CT,XT;
ENDDO;
```

In this example, the Durbin-Watson statistic is examined and an autoregressive transformation of the data is carried out if there are signs of positive autocorrelation of the residuals.

12.2 Loops over Names: DOT

The DOT statement is like a DO statement, except the index has a set of character values (names or numbers) which can be substituted into names where a dot (.) appears. For example,

```
DOT A B C ;            is equivalent to:     PRINT PA;
        PRINT P. ;                           PRINT PB;
ENDDOT ;                                     PRINT PC;
```

This makes it possible to repeat one or more statements for each of the sectors in a multisectoral body of data. To do this, the names of the series must all have the form of a generic part, common for that series across all the sectors, and a sector part, common across all the series for that sector. This is a conventional way to identify multisectoral data. For example, the generic names might be DW, (rate of change of wages), U (unemployment), and DP (rate of change of prices). The sector names might be US (United States), UK (Britain), SE (Sweden), and DE (Germany). USE would denote unemployment in Sweden, and DPDE would be the rate of change of prices in Germany.

If data are set up in this way, TSP statements may be written with just the generic parts of the names, followed by a dot (.). TSP substitutes the various sector names for the dot and executes the statements repeatedly for all sectors.

The DOT statement indicates the beginning of a set of statements to be executed repeatedly in this fashion. DOT is followed by a list of the sector names. Examples:

```
DOT US UK SE DE ;

DOT 1 2 3 4;

DOT 1-4;

LIST SUBS 1-4; DOT SUBS;

LIST(FIRST=1,LAST=4) SUBS; DOT SUBS;
```

The end of the set of statements is marked with an ENDDOT; . To specify a regression of DW on U and DP for each of the four countries in the above example, use

```
DOT US UK SE DE ;
        OLSQ DW. C,U.,DP.;
ENDDOT;
```

Any number of statements may appear between DOT and ENDDOT. TSP cycles through them as many times as there are sectors in the DOT statement. Any TSP statement may contain dotted series names. Series names without dots (for example C in the OLSQ statement) are used directly and are not concatenated with the sector name.

Dots may appear any place within a variable name, including the first character, although names that contain only numbers and a dot (.) should not be used since they will be confused with real numbers. For example, .2 is illegal. A name that consists of only a dot (.) *is* allowed, however. Here is an example where the mean is removed from each of several variables:

```
DOT S E K ;
```

```
        MSD (NOPRINT) . ;     ? Note the use of a "." all by itself
        . = . - @MEAN ;
ENDDOT ;
? The variables S, E, and K are now centered around zero
? (each of them has had its mean subtracted).
```

Double dots (two DOT loops that are nested) are also possible and can be a powerful technique. The next example evaluates a set of equations for the variables R1-R4, S1-S4 and Q1-Q4. It stores the results in variables with the tag FIT:

```
DOT (CHAR=%) S R Q ;
        DOT 1 2 3 4 ;
                GENR EQ.. .FIT. ;
                ALT.. = B.% * .FIT. ;
        ENDDOT ;
ENDDOT ;
```

The new variables are SFIT1-SFIT4, RFIT1-RFIT4, and QFIT1-QFIT4; note that the dots are evaluated in order within a name. Note also the use of the CHAR option to specify which dot is meant when there are two levels of dots. In this case B.% denotes BS, BR, and BQ as the outer loop index changes.

The DOT command has several options which allow the best features of DO and DOT loops to be combined, and give more control over single-dotted names in nested DOT loops; examples are given in the *Reference Manual*.

12.3 User Procedures: PROC

Another useful way to group TSP statements together for repetitive use is the user procedure. The group is given a name, then the name can be used anywhere in the TSP program to stand for the whole group of statements.

The beginning of a procedure is marked by PROC followed by the name of the procedure. All the statements following PROC are incorporated in the procedure, up to ENDPROC (or ENDP), which marks the end. Example:

```
PROC REGAFT;
        ACTFIT @LHV,@FIT;
        COVA @LHV,@FIT;
ENDP;
```

REGAFT; could be invoked after a regression to carry out the ACTFIT comparison and print the COVA results. For example,

```
OLSQ CONS,C,GNP;
REGAFT;
```

executes the REGAFT procedure for this regression. The effect is as if the two statements in the body of REGAFT had been placed after OLSQ instead.

User procedures may have arguments. These are variable names defined in the PROC statement which stand for the actual names used when the procedure is invoked. Any number of arguments may be listed after the name of the procedure in the PROC statement. When the procedure is invoked elsewhere in the TSP program, the same number of actual variables must be listed after the procedure name. A TSP list variable may be a procedure argument. This provides a way to pass a variable number of series to a procedure for processing. The LENGTH command may be used inside the procedure to count the number of items in the list.

Each time the procedure is invoked, TSP sets up a correspondence between the argument names used in the procedure definition and the variable names specified in the invocation.

```
PROC PCNTCH X,Y;
        GENR Y=100*(X-X(-1))/X(-1);
ENDP ;
```

PROC PCNTCH computes Y as the percent rate of change of X. For example, to compute the percent change of GNP and call it GNPPCH, specify:

```
PCNTCH GNP,GNPPCH;
```

The next example is an alternative to the first example (REGAFT), and shows the use of a list as an argument:

```
PROC REGF VARS;
        OLSQ VARS;
        ACTFIT @LHV,@FIT;
        COVA @LHV,@FIT;
ENDP REGF;
```

```
LIST V1 CONS C GNP;           ? using PROC REGF
REGF V1;
```

The next procedure computes a moving average of X length LEN over the current SMPL and returns them in XMA:

```
PROC MA X,LEN,XMA ;
        LOCAL LAST LAG ;
        XMA=X ;
        SET LAST=1-LEN ;
        DO LAG=LAST TO -1 ;
        XMA = XMA+X(LAG) ;
        ENDDO ;
        XMA=XMA/LEN ;
ENDPROC ;
```

Using this procedure, the command

```
MA R,2,RMA;
```

is equivalent to

```
RMA = (R+R(-1))/2;
```

Note the use of the LOCAL statement to name some variables that are only allocated temporary storage during the execution of procedure MA. This can be convenient if variables named LAST or LAG are in your main TSP program. The LOCAL option should be used if you wish to build a library of procedures and don't want conflicts among the variable names they use and those in the programs that might use the procedures.

One TSP procedure may invoke another and this may proceed to any depth. All TSP procedures are defined at the time that the program is read and printed, and before it is executed. Therefore, the definition of a procedure may appear in a program later than the first place that it is invoked. If errors occur during the execution of a PROC, the line number where each nested PROC was called is printed to aid in diagnosing the problem.

Note: See the TSP examples on the TSP web site www.tspintl.com for many more complex PROC examples.

12.4 Statement Label and Go To Statement: GOTO

Any statement in TSP may be given a label, which is just a number placed before the command name. The normal order of execution of statements can be modified with a GOTO (or GO TO) statement. GOTO is followed by a statement label (number) and causes that statement to be executed next. Execution then proceeds in normal order from that statement. GOTO statements are most useful in conjunction with the IF statement. Together, they can control the execution of a program using the results of computations in the program.

Note: Be careful using GOTO statements to transfer to an earlier part of the program. They can cause infinite loops.

12.5 Conditional Statements: IF, THEN, ELSE

The conditional statements IF, THEN, and ELSE can be used to specify that some statements in your program are to be skipped or executed only in some situations. IF evaluates a scalar expression. Usually the next statement is THEN; followed by a statement to be executed if the scalar expression is "true" (greater than 0). The statement after that may be ELSE; followed by a statement to be executed if the expression is "false" (less than or equal to 0). In other words, the scalar expression defines a logical condition; one statement is executed if the condition is "yes" and another if it is "no". Recall that TSP has relational and logical operators that create and manipulate zero-one variables, which can be used in an IF statement.

Examples:

```
IF .NOT. @IFCONV; THEN; GOTO 100;
```

Control is transferred to statement 100 if the previous estimation procedure did not converge; otherwise GOTO 100 is skipped and the next statement after it is executed.

```
IF LAMBDA>16; THEN;
        PRINT ALPHA,BETA;
ELSE;
        PRINT LAMBDA;
```

The expression LAMBDA>16 has the value 1.0 (TRUE or “yes”) if LAMBDA exceeds 16, in which case ALPHA and BETA will be printed;

otherwise LAMBDA itself will be printed. LAMBDA *must* be a scalar, not a series (unless the sample has been set so that only one observation is included). Use the SELECT or GENR statements with logical expressions to perform conditional transformation on series (see Chapter 3).

Avoid using IF to define loops -- the DO statement (Section 12.1) is a better way to accomplish the same thing. If a collection of several statements is to be executed conditionally, the statements may be enclosed by DO and ENDDO statements (again, see Section 12.1). This avoids GOTO statements, which usually make a program difficult to read and are unnecessary in a well-written program.

12.6 Controlling Printed Output: REGOPT

TSP allows users to specify which results are to be printed in many procedures (mainly the estimation procedures). A complete table of results available from each procedure appears in the *Reference Manual*. Note that results can be stored without being printed under names beginning with @, like @RES, @FIT, @COEF, @SSR, etc.

REGOPT enables you to change the standard selection of results to be calculated and printed. REGOPT(NOPRINT) followed by a list of code names tells TSP to print everything but the results specified by the code names. The suppression of the results takes effect when the REGOPT(NOPRINT) statement appears, and may be turned off at any point by a REGOPT(PRINT) statement. REGOPT; by itself restores the default selection. The complete list of code names available is given in the *Reference Manual*, and a partial list is given in Section 13.2.2 of this manual. Here is an example:

```
REGOPT(NOPRINT) @YMEAN,@SDEV,@ARSQ,@FST;
```

causes the dependent variable mean and standard deviation, adjusted R-squared, and F-statistic to be omitted from the output in all subsequent estimation procedures.

Many estimation procedures support the TERSE option, which suppresses all the printout except for the log likelihood value, the coefficient estimates, and their standard errors. To suppress all print output, use the SILENT option (results will just be stored under their internal names). This option is especially useful when doing many repeated estimations in a Monte-Carlo simulation experiment.

13. MATRIX COMPUTATIONS

Earlier in this manual we mentioned that one of the several TSP variable types available is the matrix. This chapter shows how to create matrices, combine them with other data in your TSP program, and perform computations on them. The advantage of working with matrices in TSP is that it facilitates computation of estimators not provided as "canned" procedures within TSP. We give some examples of these estimators later in this chapter.

Here is a simple application involving matrices that illustrates how they can be used to compute a residual covariance matrix:

```
OLSQ INV1 C F1 K1 ;
COPY @RES E1 ;              ? Save residuals from equation (1).
OLSQ INV2 C F2 K2 ;
COPY @RES E2 ;              ? Save residuals from equation (2).
MMAKE E E1 E2 ;             ? Make a T by 2 matrix of residuals.
?    Form the 2 by 2 covariance matrix of these residuals:
MAT RESCOV = (E'E)/@NOB ;
PRINT RESCOV ;              ? Print the matrix.
```

In the above example, we form an estimate of a asymptotic residual covariance matrix for a two-equation model estimated by ordinary least squares. E1 and E2 contain the residual series for each equation; each can be thought of as a T (where T is the number of observations) by 1 vector. MMAKE combines the two series to make a T by 2 matrix:

```
E = [ E1 E2 ]
```

The next command, MAT, is a matrix computation command. It calculates the 2 by 2 matrix RESCOV as the cross-product matrix of the residual matrix E divided by the number of observations in the regressions. This matrix is an estimate of the covariance of the disturbances in the two equations. The result is displayed by PRINT. PRINT can be used for matrices as well as for variables of any other type in TSP.

This simple example introduces combining matrix operations with other information in your TSP program. Before giving any more examples, we give an overview of the matrix facilities available in TSP: the types of matrices available, how to create or input matrices, and how to compute using matrices .

13.1 Matrix formats

To economize on matrix storage and computation, TSP stores matrices of several different types in different ways, and uses only the unique and non-zero elements of a matrix to save on computations. For the most part, you need not be aware of this; the program takes care of it for you. When you load matrices, however, it is sometimes helpful to label the matrix type to save data entry. You will also notice that the format of the printed output of matrices will vary by matrix type. The matrix types recognized by TSP are the following:

GENERAL -- A general matrix may have any rectangular form (any number of rows and columns). Any matrix can be loaded or used as a general matrix, even one that has a special form. A vector has only one row or only one column. Some matrix operations allow series to be used as vectors (provided that the number of observations in the current sample conforms to the other matrices in the procedure). When they are used in matrix procedures, scalars are treated as special cases of general matrices with number of rows and columns equal to one.

TRIANG -- A triangular matrix contains meaningful elements on and above the diagonal and zeroes below the diagonal. TSP stores only the diagonal and above diagonal elements. This corresponds to an upper triangular matrix; if you wish a lower triangular matrix, use the transpose of this matrix in all your operations. When a triangular matrix is printed out, elements below the diagonal are blank.

SYM -- A symmetric matrix has the same elements above the diagonal as below the diagonal. TSP stores only the elements below the diagonal and expands the matrix before it is used. When the matrix is printed out, the elements above the diagonal are blank.

DIAG -- A diagonal matrix is any square matrix with nonzero or zero elements on the diagonal and zero elements everywhere off the diagonal. Diagonal matrices are obviously also triangular and symmetric. TSP stores only the diagonal of this type of matrix, and expands the matrix before it is used. It can be created or reformed from a vector (the diag operation from matrix algebra) using the procedure MFORM or the matrix function DIAG( ). If GAMMA is a vector of length 5, examples showing how to form the matrix diag(GAMMA) are the following:

```
MFORM(NROW=5,TYPE=DIAG) DG=GAMMA ;
```

or

```
MAT DG = DIAG(GAMMA) ;
```

The resulting 5 by 5 diagonal matrix DG has zeros everywhere except along the main diagonal, where DG(1,1) = GAMMA(1), DG(2,2) = GAMMA(2), and so forth.

An identity matrix may also be formed in this way, but it is more easily formed with the IDENT() function:

```
MAT ID = IDENT(5) ;
```

13.2 Creating a matrix

There are three ways to create a matrix in a TSP program: read it in with your input data, obtain it as the result of a TSP procedure, or form it from a group of time series or other data. The next three sections describe each of these methods.

13.2.1 Reading matrices: READ

Matrices can be read just like series with two important differences: the current SMPL does not apply to them and options on the READ command tell the program the matrix's format. These options are the matrix type (TYPE=GENERAL, SYMMETRI, TRIANG, or DIAG), number of rows (NROWS=), number of columns (NCOL=), and whether the full matrix or just the significant elements are being loaded (FULL/NOFULL).

Here is an example of reading a general matrix:

```
READ (NROW=4,NCOL=3) COEFMAT ;
.32 0.5 1.3
.30 0.4 1.35
.25 0.61 1.1
.28 .55 1.23
;
```

The data should consist of all the elements of the first row followed by all the elements of the second row, and so on, with a ";" at the end of the last column.

In this example, we entered each row of the matrix on a separate line to make it easy to read and to check, but this is not required.

The next two examples of reading a symmetric matrix produce the same result; they demonstrate the FULL option.

```
READ (NROW=3,TYPE=SYM) COVAR ;
4.6
2.3 5.1
0.9 2.1 3.9 ;

READ (NROW=3,TYPE=SYM,FULL) COVAR ;
4.6 2.3 0.9
2.3 5.1 2.1
0.9 2.1 3.9;
```

The FULL option is used mainly when the matrix is already in this format because it was obtained from a source other than TSP, and you do not want to remove the redundant elements.

Matrices do not have to be read in free format, as they were in these examples; you can use any of the formats described for the READ command in the *Reference Manual* or in Chapters 3 and 16 of this manual.

13.2.2 Matrix results from TSP procedures: COPY

The second way to create a matrix is to obtain it from the results of a matrix procedure or a statistical procedure. For example, to use the estimated covariance matrix from an estimation procedure later in your TSP program, the command

```
COPY @VCOV Q;
```

will save the matrix for later use under the name Q.

All results which can be obtained in this way are listed in the *Reference Manual* or online *Help System* under the Output Section for each procedure. The exact dimensions and type of matrices concerned are also given. Most of the results available are printed by the estimation procedures (OLSQ, 2SLS, LIML, AR1, LSQ, FIML, etc.) and are only available for use until the next estimation procedure, since the same temporary names (beginning with @) are reused for all procedures.

Some of the standard matrix results stored after estimation procedures are:

@COEF	the vector of coefficient estimates
@SES	the vector of standard errors of the coefficients
@VCOV	the estimated variance-covariance matrix
@VCOR	the correlation matrix corresponding to @VCOV[24]
@LAGF	the vector of estimated lag coefficients (if a PDL variable was used)

All the results below are stored as vectors rather than scalars if a multi-equation model was estimated, so they may be manipulated as matrices:

@SSR	the sum of squared residuals
@S2	the estimated variance of the residuals
@S	the standard error of the residuals
@RSQ	the R-squared
@DW	the Durbin-Watson statistic
@YMEAN	the mean of the dependent variable
@SDEV	the standard deviation of the dependent variable

These additional results are stored as scalars for the single equation models only:[25]

@FST	the F-statistic
@ARSQ	the adjusted R-squared or R-bar-squared
@DH	Durbin's h statistic (for a lagged dependent variable)
@DHALT	Durbin's alternate statistic (for lagged dep. variables)
@LOGL	Log of likelihood function
@AIC	Akaike Information Criterion
@SBIC	Schwarz Bayesian Information Criterion
@LMHET	Lagrange multiplier test for heteroskedasticity
@RESET2	Ramsey's RESET test of functional form
@JB	Jarque-Bera statistic

Residuals @RES and fitted values @FIT are stored as matrices (rather than

[24] REGOPT(CALC) VCOR; must be in effect for this to be retrievable.

[25] LSQ, GMM, and FIML do not store @YMEAN, @SDEV, @FIT, @RSQ, etc. for unnormalized equations (those without explicit dependent variables).

series) if a multi-equation model is being estimated. The number of rows in these matrices is equal to the number of observations and the number of columns is equal to the number of equations.

The multi-equation estimation procedures (VAR, LSQ, SUR, GMM, 3SLS, FIML) also store:

@COVU	the covariance matrix of the residuals.

Most nonlinear procedures (LSQ, FIML, PROBIT, ML, etc.) store:

@IFCONV	convergence flag (1=converged, 0=not converged)
@GRAD	gradient vector w.r.t. the parameters at convergence

The procedures for simple statistics, MSD, CORR, COVA, and MOM, also store their results as matrices. Here is a partial list of useful results:

@MEAN	Vector of series means
@STDDEV	Vector of series standard deviations
@MIN	Vector of series minimums
@MAX	Vector of series maximums
@SUM	Vector of series sums
@VAR	Vector of series variances
@COVA	Covariance matrix

See the *Reference Manual* or online *Help System* for a complete list of results available in each procedure.

13.2.3 Creating a matrix: MMAKE, UNMAKE

The third way to create a matrix is to assemble one from a set of time series or scalars. MMAKE makes each series into a column of a matrix. To use this command follow the name of the matrix by the names of the series that will go into the matrix. The current SMPL controls the selection of observations from the series. For example, if the following series had been loaded,

```
SMPL 1,3;
LOAD X; 1,2,3;
LOAD Y; 4,5,6;
LOAD Z; 7,8,9;
```

then the program could have

```
MMAKE M X,Y,Z;
```

which would create the matrix M shown below:

```
         1    4    7
   M =   2    5    8
         3    6    9
```

To stack the three series into a 9x1 vector, you could use MFORM to reshape M:

```
MFORM(NROW=9,NCOL=1) S=M ;
```

Alternatively, you could use the VERT option on MMAKE:

```
MMAKE(VERT) S X,Y,Z ;
```

Either way, the matrix S = [1 2 3 4 5 6 7 8 9]’ would be created.

The inverse of MMAKE is UNMAKE. To store the columns of a matrix as individual series, use UNMAKE followed by the name of a matrix and then the names for the new series. The operation must be conformable; that is, the number of rows in the matrix must be equal to the number of observations in the *current* SMPL and the number of columns must be equal to the number of series names given.

Below is an example that undoes the first MMAKE X,Y,Z:

```
UNMAKE M,A,B,Q;
```

After execution, A,B, and Q are time series that are identical to the original X, Y and Z.

To turn a matrix back into a series, use UNMAKE with a single series name. For example, to create a stacked series T after stacking using MMAKE (VERT),

```
SMPL 1, 9 ;
```

```
UNMAKE S T ;
```

MMAKE and UNMAKE also operate on scalar variables; in this case the first argument is a vector. This is useful for operations on starting values or parameter estimates. For example,

```
OLSQ Y C X1-X10;
PARAM B0-B10;
UNMAKE @COEF B0-B10;
```

places the OLS estimates of the coefficients into the scalars B0, B1, B2, ... for further use.

MMAKE can also be used to concatenate matrices. For example, suppose that you have matrices Z1 (3 by 2), ZIT (2 by 3), and Z2 (3 by 3). The command

```
MMAKE NEWZ Z1 Z2 ;
```

makes the 3 by 5 matrix

NEWZ = [Z1 Z2]

The command

```
MMAKE(VERT) NEWZT ZIT Z2 ;
```

makes the 5 by 3 matrix

$$NEWZT = \begin{bmatrix} Z1T \\ Z2 \end{bmatrix}$$

13.2.4 Making a matrix from other matrices: MFORM

We have already given a few examples of the use of MFORM to copy or reformat matrices. It can also be used to change the type of matrix (from symmetric to general, for example):

```
MFORM(TYPE=GEN) YMAT ;
```

The above example takes YMAT, a symmetric matrix stored as a lower triangle, and fills out the elements above the diagonal in the appropriate way, storing the matrix under the same name. To change its name to GMAT at the same time, use

```
MFORM(TYPE=GEN) GMAT = YMAT ;
```

MFORM can also be used to create block diagonal matrices from other matrices; for example, the command

```
MFORM(BLOCK) NEWMAT = OLDMAT1 OLDMAT2 OLDMAT3 ;
```

forms the matrix NEWMAT from three matrices (all of which are normally square matrices, but need not be) using the block diagonal format. If OLDMAT1 is 2 by 2, OLDMAT2 is 3 by 3, and OLDMAT3 is 4 by 4, the new matrix NEWMAT will be a 9 by 9 general matrix, with zeroes in the appropriate locations.

13.3 Matrix algebra: MAT

In TSP you can perform algebraic operations with matrices just the way you can with series, using the same kind of easily specified formulas or equations. For example, the command

```
MAT B = (X'X)"X'Y ;
```

computes a vector of regression coefficients and stores them in B, given a matrix of data for the regressands in X and a vector of data for the regressor in Y. Note the use of the operator ' (apostrophe) to specify the transpose of X, and the operator " (quotation mark) to specify the operation of matrix inversion. Of course, you are unlikely to use this particular matrix equation, since TSP performs this operation automatically when you specify an OLSQ command, which uses a more accurate algorithm. To use OLSQ to compute this expression without creating print output, use the following:

```
UNMAKE X X1 ... XK ;
OLSQ (SILENT) Y X1 .... XK C ;        ? X1, etc are columns of X
COPY @COEF B ;
```

In the next few sections, we describe in more detail how to perform calculations using matrices. Matrix operations basically fall into three classes: ordinary operators (multiplication, addition, etc.), functions of matrices that yield scalars or matrices as output, and matrix procedures which have two or more arguments. Only the latter cannot be included in matrix equations specified in the MAT command and must be specified using separate commands. We begin with the operations and functions that can be used with a MAT statement, and then describe the stand-alone procedures.

13.3.1 MAT command and matrix operations

TSP provides a variety of operations on matrices, including multiplication, inversion, factorization, calculation of eigenvectors, and eigenvalues, and so forth. All these operations may be specified in matrix equations preceded by the word MAT; these equations are just like the variable transformations performed by GENR, except for two things: they do not operate under control of the current SMPL and the result from MAT equations is stored as a matrix. The MAT procedure checks the matrices for conformability of the operations and gives an error message if the operation specified is not possible. Often printing the matrices in question will reveal why the operation cannot be performed.

Appendix A gives a complete list of TSP operators and what they mean. In this section we list these operators and how they are interpreted in the MAT command. All the ordinary operators and functions used in TSP equations can also be used in the MAT command. They operate on an element-by-element basis (and hence require conforming matrices if they are binary operators). There is one important exception to this, the multiply operator * . For simplicity, this operator denotes the usual matrix multiplication, and element-by-element multiplication (the Hadamard product) is denoted by the operator % .

In the descriptions of the matrix operators that follow, we use the following symbols to denote the inputs and outputs of operations:

Figure 13.1 Argument types for MATRIX

Symbol	*Description of variable*
s	Scalar or subscript variable
i	Integer scalar
m	Any matrix (if scalar, treated as 1 by 1 matrix)
qm	Square matrix, N by N
sm	Symmetric matrix, assumed positive semi-definite
dm	Diagonal matrix, assumed positive semi-definite
tm	Upper-triangular matrix, assumed positive semi-definite
v	Column vector, N by 1

The additional symbolic operators understood by the MAT are shown below. Remember that the operands *must* be conformable for the operations that you request; TSP will check the dimensions for you and refuse to perform the computation if this condition is violated.

Figure 13.2 Matrix operators

Operation	*Description*
m*m	Matrix product
m*s or s*m	Scalar multiplication
m’	Matrix transpose
m’m	Matrix transpose with implied matrix product
qm”	Matrix inverse (also causes @DET, the determinant of *qm*, to be stored)
qm”m	Matrix inverse with implied matrix product
m#m	Kronecker product
m%m	Hadamard product (element by element multiplication)

When TSP processes a MAT command, it recognizes several operations where great savings of computation time can be made by eliminating duplicate calculations. These situations include, but are not limited to, the cross-product operation (which generates a symmetric matrix) and the calculation of a quadratic form (the expression A*B*A'). This occurs even when the arguments to these expressions are complicated expressions themselves. Thus, you should be careful to express any such complex arguments in the same way whenever they appear in the matrix expression. For example, the following matrix equation computes the well-known test for a set of linear restrictions on estimated regression coefficients (H_0: $Rb = b_0$):

```
MAT CHI = (R*B-B0)'(sigsq*(R*(X'X)"R'))"(R*B-B0) ;
```

Both quadratic forms in this expression will be recognized and computed as such, and the expression R*B-B0 will be computed only once.

13.3.2 Matrix functions with scalar output

The following functions take matrices as their input and produce scalars as output:

Figure 13.3 Matrix functions – scalar output

Function	***Output type***	***Description***
DET(*qm*)	*s*	Determinant (truncated to zero when <1.E-37)
LOGDET(*qm*)	*s*	Log of (positive) determinant, no truncation
TR(*qm*)	*s*	Trace (sum of diagonal elements)
MIN(*m*)	*s*	Element with minimum value
MAX(*m*)	*s*	Element with maximum value
SUM(*m*)	*s*	Sum of elements
NROW(*m*)	*i*	Number of rows
NCOL(*m*)	*i*	Number of columns
RANK(*m*)	*i*	Rank (number of linearly independent columns or rows)

These functions may be used anywhere in a MAT statement where scalars are allowed, keeping in mind that a scalar is also a 1 by 1 matrix.

13.3.3 Matrix functions with matrix output

The following functions are matrix-to-matrix; that is, they take a matrix, perform some computation on it, and produce another matrix as output:

Figure 13.4 Matrix functions – matrix output

Function	***Output type***	***Description***
CHOL(*sm*)	*tm*	Choleski factorization (matrix square root)
YINV(*sm*)	*sm*	Positive semi-definite inverse using CHOL()
IDENT(*i*)	*dm*	Creates an identity matrix of order *i*
EIGVAL(*qm*)	*v*	Computes the vector of eigenvalues of *qm*. If qm is not symmetric positive semi-definite, the imaginary parts of the eigenvalues are stored as @EIGVALI.
EIGVEC(*qm*)	*qm*	Computes the matrix of eigenvectors (columns). If *qm* is not symmetric positive semi-definite, the imaginary parts of the eigenvalues are stored as @EIGVECI. @EIGVAL and @EIGVALI will also be stored automatically. EIGVAL and the corresponding EIGVEC are sorted high to low
VEC(*m*)	*v*	Creates a vector of all the elements of *m,* column by column.
VECH(*m*)	*v*	Creates a vector of all the unique elements of *m*, column by column: *qm* N*N elements *sm,tm* N*(N+1)/2 elements *dm* N elements
DIAG(*m*)	*dm*	Creates a diagonal matrix from a matrix: *qm, sm, tm* take the diagonal from input matrix *v* convert the vector to a diagonal matrix *s* illegal; use s*IDENT(i) to create a diagonal matrix
SYM(*qm*)	*sm*	Creates a symmetric matrix from a square matrix (the upper triangular elements are ignored).
GEN(*qm*)	*m*	Creates a general matrix from a symmetric or diagonal matrix.
SER(*m*)	*series*	Creates a series from a vector (same as UNMAKE v series ;). If used this function must be the outermost (last) function, and the length of *v* must match the number of observations in the current SMPL.

13.3.4 Matrix procedures: ORTHON, YLDFAC

Several matrix procedures have more than one output and therefore cannot be used in MAT. These procedures are YLDFAC for performing an LDL decomposition (factorization of a semi-definite matrix), and ORTHON for matrix orthonormalization. You can use them like an ordinary TSP command with arguments; the input arguments are the names of matrices and the output arguments will be stored as matrices.

There are two procedures available for factorization (generalized square root) of symmetric matrices: the CHOL function performs a Choleski decomposition, and YLDFAC performs an LDL decomposition.

```
MAT S = CHOL(A) ;
```

finds an upper triangular matrix S such that A = S'S. If A is a symmetric positive semi-definite matrix, there must exist such an S. The number of nonzero elements on the diagonals of S will be equal to the rank of A. Note also that $A^{-1} = S^{-1}(S')^{-1}$.

```
YLDFAC A D U ;
```

finds a diagonal matrix D and an upper triangular matrix U such that A = U'DU. This decomposition always exists if A is symmetric; the diagonal elements of D are functions of the eigenvalues (characteristic roots) of A. The number of nonzero diagonals of D is the rank of A, and the sign of the diagonals indicate whether the matrix A is positive definite (all positive), negative definite, or indefinite.

If DHALF is a diagonal matrix of whose elements are the square roots of the corresponding elements of D, then the relationship between S and U is S = DHALF*U. This makes it obvious that S is only defined when the elements of D are non-negative.

Orthonormalization is a transformation of a data matrix so that its columns are orthogonal and have a unit norm. This transformation is used by TSP's regression calculation to improve the accuracy of the results. The appropriate transformation is obtained by forming the cross product of the data matrix, X'X, factoring it to obtain the triangular matrix S, inverting S and using it to transform the original data:

$$Z = X\,S^{-1}$$

Since X'X = S'S, the resulting matrix Z will have the property

$$Z'Z = S^{-1}{}'\,X'X\,S^{-1} = I,$$

that is, the columns of Z will be orthonormal. The form of the command is

```
ORTHON X SINV Z ;
```

where X is the input matrix, SINV is the name to give its inverse square root, and Z is the name to give the orthonormalized X.

13.4 *Examples using matrix operations*

The best way to learn how to use TSP matrix operations is to look at examples, as it is not always obvious how to do what you want. This section presents some solutions to common computation problems not yet implemented in TSP procedures. They include performing a Hausman specification test, computing the prediction error for a simple linear model, and computation of a ridge estimator for the linear regression model.

13.4.1 A Hausman specification test

A Hausman test compares two sets of estimates of the same parameters using the same data: one obtained using an efficient estimation technique assuming the specification is correct, and another obtained by an estimation method that is consistent even under a set of alternate hypotheses about the specification. The test is computed by differencing the two sets of parameter estimates and standardizing the vector of differences by the difference in the covariance matrices of the two sets of estimates. The quadratic form computed this way is asymptotically chi-squared with degrees of freedom equal to the number of parameters being tested (with certain caveats that we will mention later).

To compute this test in TSP, we assume that somewhere you have obtained efficient estimates of the parameters, BEFF, and their variance, VEFF. Least squares or some other regression procedure is used to obtain consistent estimates of the parameters under a wide variety of misspecifications of the error term; these estimates are obtained immediately before you perform the test so they will be named @COEF and @VCOV. The Hausman test is computed by the following matrix operations:

```
MAT DVAR = @VCOV-VEFF ;
MAT K = RANK(DVAR) ;
MAT HTEST = (@COEF-BEFF)'YINV(DVAR)*(@COEF-BEFF) ;
CDF(CHISQ,DF=K) HTEST ;
```

The order of subtraction is important for the variance estimates, since the asymptotic variance of the efficient estimate must be less than or equal to that of the consistent estimate (in the matrix sense). However, frequently in practice, some elements of DVAR are negative on the diagonal. If this is the case, the test should be computed only for those parameters corresponding to positive diagonal elements, with a corresponding correction to the degrees of freedom. The program shown above does this automatically, since it sets the degrees of freedom equal to the rank of DVAR. The matrix inversion function YINV() will automatically perform a generalized inverse that sets the linearly dependent rows and columns of the inverse to zero.

13.4.2 Prediction error for the linear regression model

When regressors in the classical linear regression model are fixed constants and the disturbances outside the sample are assumed to be drawn from the same distribution as those within, the variance of the predictor from this model has two pieces. The first is due to the variance of the disturbance and the second to the variance of the estimate of the coefficients. If X is the matrix of in-sample regressors, X_0 is the data for the prediction sample, and s^2 is the variance of the disturbances, the equation for the prediction error variance is

$$V = s^2 \left[I + X_0 (X'X)^{-1} X_0' \right]$$

You can use this formula to compute standard error bounds for a forecast of a linear regression model in TSP using matrix operations. Suppose that you have run an OLSQ for the historical period 1958 to 1982 and now wish to forecast through to 1990 using values of the exogenous variables previously projected. The forecast itself can be computed (without printing in this case) immediately following the regression:

```
SMPL 58,82;
OLSQ Y C X1 X2;
SMPL 83 90 ;
FORCST YPRED ;
```

Now use the following matrix commands to construct an estimate of the prediction error:

```
? Make a matrix of the data for the forecast time period:
MMAKE X @RNMS ;
? Compute the prediction error.
MAT PREDERR = SER(SQRT(@S2+VECH(DIAG(X*@VCOV*X')))) ;
? Plot the forecast with standard error bands for the prediction error:
PLOT(BAND=PREDERR) YPRED ;
```

Note the use of the SER function to convert the output of the computation into a series, and the VECH and DIAG functions to extract the diagonal elements of X*@VCOV*X'.

13.4.3 Ridge regression

Ridge regression is a Bayesian estimator of a linear regression model, with a prior of zero on all coefficients. Ridge estimators have also been viewed by some observers as a way of overcoming multicollinearity in the independent variables. For a discussion of these estimators, see Judge et al.(1980), pp. 452-501. A ridge estimator of the coefficients β from a linear regression model

$$y = X\beta + \varepsilon$$

has the form

$$b = (X'X + \gamma I)^{-1} X'y$$

where γ is a positive scalar (which can be computed in a number of ways) and I the identity matrix of order of the number of independent regressors. It can be seen easily that the effect of adding a positive constant to all the diagonal elements of $X'X$ will be to reduce the tendency for $X'X$ to be singular or nearly singular.

The example we give of computing a ridge estimator in TSP uses a formula for the scalar γ suggested by Sclove (1973) and described in Amemiya (1985). It is an empirical Bayesian estimator. The prior is that the coefficients are zero with a variance estimated from the data as the sums of squared of the fitted values of y divided by the trace of the $X'X$ matrix. The γ coefficient in this case is a consistent estimate of the residual variance divided by the variance of the coefficient prior. The example below forms an estimate of γ by performing a conventional OLS regression, computing estimates of the variances, and using the matrix algebra routines to run the ridge regression.

```
? Ordinary least squares on original data.
```

```
OLSQ Y C X2 X3 X4 X5 ;
COPY @COEF ALPHA ;

? Compute the Sclove coefficient gamma.
MMAKE X @RNMS ;
MAT GAMMA = (@SSR/@NOB)/((Y'Y-@SSR)/TR(X'X)) ;

? Now form the estimator
MAT AHATSTAR = (X'X+GAMMA*IDENT(@NCOEF))" X'Y ;
PRINT GAMMA ALPHA AHATSTAR ;
```

14. FORECASTING AND MODEL SIMULATION

Often the primary goal of an econometric study is to produce a model for forecasting an economy's future behavior, a sector in an economy, or even the behavior of an individual firm. Two common approaches to this problem are the structural model method and the time series (ARIMA or VAR) methods. Both methods can be used in TSP. A discussion of the pros and cons of these two methods is beyond the scope of this manual; an excellent reference that describes both methods, emphasizing their forecasting aspects is the Pindyck and Rubinfeld text.

Chapter 11 introduced the time series methods of forecasting variables using ARIMA or VAR models. In this chapter, we discuss how to use the structural model forecasting methods in TSP. First we give an overview of the steps in constructing a forecasting model of any type, then we describe methods for forecasting a single variable, and finally we describe the model solution procedures used for multi-equation models.

There are usually three major steps in producing a forecast:

1. Define a model with one or more equations giving a reasonably good description of the variables of interest over a recent time period. The model can come from any source, although multi-equation models must be expressible in TSP equations (FRMLs). In particular, the model may arise from a series of estimations within TSP, both nonlinear and linear. The dimensions and quality of the model are your choice. If you want to use ARIMA models, use BJIDENT to help you choose the model.

2. Choose a forecast period and make some assumptions about the behavior of the exogenous variables in the model over this period. In an ARIMA forecast, there are no exogenous variables in the model, so you can skip this step.

3. Using projected values of the exogenous variables and the model you have chosen, choose one of TSP's forecasting or simulation procedures to compute the endogenous variables of the model one period at a time. Lagged endogenous variables may be determined from either the previous forecasts (dynamic simulation -- this method will always be used for ARIMA or VAR forecasts), or may be actual realizations of the variable (static or historical

simulation).

TSP provides several procedures and techniques to compute the forecast and display the results. These techniques can be divided into two major groups, single equation (single variable) forecasts, and the solution of simultaneous equations models. We describe the former first, and then describe the model solution procedures, SIML and SOLVE.

14.1 Creating equations: FRML, FORM

Chapter 7 described how FRML can be used to make TSP equations. Any equation thus created can be used in forecasting with the proviso that equations for SOLVE must be normalized (have a left-hand side variable). Note that logical expressions like **Y>0** may also be included in the equations of a simulation model.

It is also important to remember that equations used in simulation must have unknown parameter values defined in some way beforehand. This can be done by using a SET or PARAM statement to supply a value, for example,

```
FRML GNPGROW GNP = (1+ALPHA)*GNP(-1) ;
SET ALPHA = .15 ;
```

You can also estimate the equation earlier in your TSP program; in this case, the parameter will retain its estimated value when the simulation is performed. For example, using the same equation for GNP growth as above,

```
PARAM ALPHA ;
LSQ GNPGROW ;   ? at the conclusion of the estimation, ALPHA
                ? will contain its estimated value.
```

Databanks are also useful for saving parameter values after an estimation. If the estimated value of ALPHA were 0.123, the equation GNPGROW would now be interpreted by GENR, SIML, or SOLVE as

```
FRML GNP = 1.123*GNP(-1) ;
```

FRML works well when estimating nonlinear models, but many equations will be estimated by linear methods, so it may be convenient to construct linear equations automatically. FORM does this easily:

```
OLSQ CX C W P P(-1) ;
FORM CONS ;
```

This example estimates a consumption equation by ordinary least squares and forms an equation named CONS as though you had entered the following FRML statement:

```
FRML CONS CX=16.6+.81*W+.017*P+.216*P(-1);
```

[assuming that the coefficient vector estimated in the OLSQ procedure was (16.6, .81, .017, .216)].

FORM may be used after any linear estimation procedure: OLSQ, 2SLS, LIML, LAD, or AR1. If it is used following an AR1 estimation, the term involving the estimated ρ times the lagged residual is automatically added (see Section 14.3). FORM can also be used to create unnormalized equations and to create equations with parameter names rather than values. See the *Reference Manual* for details.

14.2 Forecasting with an explicit equation: GENR

The simplest way to forecast a single variable is with GENR, if an explicit equation has already been defined. For example, the following two sets of statements are equivalent in their result:

```
SMPL 83 90 ;
GENR GNPFIT = A0 + A1*POP + A2*IMPT ;

SMPL 83 90 ;
FRML GNPEQ GNP = A0 + A1*POP + A2*IMPT ;
GENR GNPEQ GNPFIT;
```

This method even works when the variable to be forecast depends on its own lagged values (which is described below), because GENR will operate dynamically. The alternate method for computing single equation forecasts, FORCST, can be used for either static or dynamic simulation. In addition, FORCST does not require that you specify the equation beforehand; it can be computed directly from a previous linear estimation.

14.3 Forecasting linear models: FORCST

FORCST will do a forecast immediately following any single equation linear estimation command in TSP (OLSQ, 2SLS, LIML, LAD, or AR1) without requiring FORM or a FRML. This also works on PROBIT and TOBIT models, although you may wish to transform the predicted latent variable.

Here are some examples:

```
FREQ A ; SMPL 48 56 ;
OLSQ IMPT C GNP RELP ;
FORCST(PRINT) IMPFIT ;
```

In this example, the forecast is computed over the estimation sample and therefore IMPFIT is simply the fitted values of IMPT (equal to the automatically stored @FIT series). It is generally true that if you compute a static forecast over the same sample as the regression, the output of FORCST will be the fitted values from the regression.

```
FREQ A ; SMPL 48 82 ;
AR1 (METHOD=CORC,PRINT) IMPT C GNP RELP ;
SMPL 83 90 ;
FORCST(PRINT) IMPFOR ;
```

This example illustrates the computation of a dynamic forecast when the errors are serially correlated. Projected values of GNP and RELP must exist for the period 1983 to 1990; the simulation uses the last historical period (1982) to calculate a presample residual. For this kind of model, the forecasting equation is

$$y_t = X_t b + \hat{\rho} e_{t-1} \quad \text{where} \quad e_{t-1} = \hat{\rho} e_{t-2}$$

The initial condition for *e* is the presample residual.

```
FREQ A ; SMPL 48 82 ;
OLSQ IMPT C IMPT(-1) GNP ;
SMPL 83 90 ;
FORCST IMPDYN ;
```

This example shows forecasting when there is a lagged dependent variable in the model. Since IMPT has been referred to explicitly with a lag on the right-hand side of the original equation, the dynamic forecast will automatically use the predicted values of IMPT in the next period's forecast. To use this feature, you must refer to the lagged endogenous variables as IMPT(-1); use of a computed variable, such as IMPTL1 = IMPT(-1), will not have the same effect.

It is also possible to use FORCST independently of any estimation command; in this case you must explicitly supply the procedure with all the information it needs to compute the forecast:

```
FORCST(DYNAM,COEF=BETA,RHO=R,DEPVAR=IMPT) IMPFIT C
        GNP RELP ;
```

In this example, BETA is the vector of estimated coefficients of the independent variables, R the estimated value of the serial correlation coefficient, IMPT the name of the original dependent variable (needed for dynamic forecasts), IMPFIT the name to be given to the forecasted variable, and C, GNP, and RELP are the independent variables in the forecasting equation. The coefficients BETA must be supplied in the same order as the list of independent variables.

14.4 Solving simultaneous equation models

TSP provides two quite different procedures for the solution of simultaneous equation models. The methods differ in their speed of convergence, use of computer storage and time, and ability to handle highly nonlinear or very simultaneous models.

The most general and powerful procedure is SIML, which uses Newton's method applied to nonlinear equation solution. The model is specified just like FIML. SIML does not require normalized equations nor that the model be ordered in a particular way, and uses analytic derivatives of the model in the solution process. For linear models, SIML converges in one iteration, and is very fast for multiperiod models. For highly nonlinear models, Newton's method (Gauss-Newton) may be the only method in TSP that can provide a solution. The cost of this power is considerable use of computer storage compared with other model solution techniques. Consequently, SIML may not be the method of choice for more than 50 equations. TSP does not explicitly limit the number of equations, but you may not have enough memory available. You may also be limited by TSP’s (relatively large) limits on the number of unique variable names/arguments in the collected model and on the number of variables in a TSP session. Use the SHOW command to see what these limits are in your version of TSP.

For large economic models, particularly sparse models with some sort of block structure, SOLVE will be more suitable. To use this procedure, first collect the equations of the model using MODEL to determine the best order for solution. The equations must be "normalized", i.e., each endogenous variable must appear once and only once on the left-hand side of an equation. MODEL determines how the equations are to be arranged into recursive and simultaneous blocks for solution. SOLVE will evaluate the recursive blocks (blocks in which every equation uses only previously computed endogenous variables as input) and will solve the simultaneous blocks either by the Gauss-Seidel method or by the more powerful Fletcher-Powell method. This method of model solution is suitable for most large economic models, which tend to be sparse, fairly linear, and separable into blocks. (It has been used on models with more than 400 equations.) However, convergence of the simultaneous blocks is not guaranteed, since it does not compute any analytic derivatives.

Details on using these two procedures are given below.

14.4.1 Small nonlinear models: SIML

SIML invokes Newton's method; it has the same form as FIML described in Chapter 7: SIML, followed by options including ENDOG=(list of endogenous variables), followed by a list of equations, including identities. Unlike FIML, SIML treats IDENTs the same as FRMLs (see Section 7.1). For example, the solution of the illustrative model is specified by

```
SIML(ENDOG=(GNP,CONS,I,R,LP))
        CONSEQ,INVEQ,INTRSTEQ,GNPID,PRICEQ;
```

The default values of the options are Newton's method, no storage of the results, and a dynamic simulation. A static simulation is often used when simulating over a historical period: it uses actual realized values of the lagged endogenous variables rather than obtaining them from the simulation of the previous period. The default for printing is to print a one line summary of each iteration and a table of the solved variables at the end.

An example of SIML that stores the results and prints the input data is

```
SIML(PRNDAT,TAG=S,ENDOG=(GNP,CONS,I,R,LP))
        CONSEQ,INVEQ,INTRSTEQ,GNPID,PRICEQ;
```

After this model has been solved, the solved series are stored under the names GNPS, CONSS, IS, etc. Further details on the options available with SIML are given in the *Reference Manual*.

Newton's Method

The method used by SIML for solution of nonlinear simultaneous equation models is a variant of Newton's method. Obviously, if a simultaneous model is linear, it can be solved directly by matrix inversion. That is, the solution to the matrix equation

$$A\,x = b$$

is

$$x = A^{-1}b.$$

Newton's method applies this idea to the iterative solution of nonlinear models. At each iteration, the model is linearized in its variables around the values from the previous iteration. The linearized model is solved by matrix inversion. In terms of the equation above, A is the Jacobian of the model with respect to the variables, b the vector of model values at the previous iteration and x the direction vector of changes in the variables to be computed.

The resulting vector of changes is used as a direction vector in a linear search for better values of the variables, as in the other iterative procedures such as LSQ. A deviation is computed for each equation by substituting the current values of the variables; at the solution, all the deviations will be zero. The criterion for the search is the sum of squared deviations of the equations; it is printed as F = (the initial value) and FNEW = (the value after the iteration) in SIML's output. Unless a solution has been achieved, there will be a better set of values somewhere along the direction vector. The actual choice of the new set of variable values is made by the same methods used in nonlinear estimation and is described in Chapter 10. For a further description of Newton's method, see Saaty and Bram (1964) or Ortega and Rheinboldt (1970).

The POS function can be used to constrain the solved values to be non-negative.

14.4.2 Large models – MODEL and SOLVE

Large models require two steps (procedures) for solution -- MODEL and SOLVE. MODEL is used to group the equations into smaller blocks, and

SOLVE seeks the solution using this structural information. Several SOLVE commands can use the same MODEL (using different scenarios of exogenous variables or parameter values, for example). SOLVE uses the same iteration and TAG options as SIML.

Ordering equations: MODEL

MODEL groups the equations of a model into blocks and saves this structure under a model name. We use the basic Klein Model I as a simple example. This model consists of three behavioral equations and four identities. There are seven endogenous variables (CX, I, W1, P, K, W and E), three lagged endogenous variables (P(-1), K(-1), and E(-1)), and four exogenous variables (G, TIME, TX, and W2). The equations may be estimated by a single equation method, using FORM to construct a TSP equation, or they may be specified with FRMLs and estimated with multi-equation methods. We assume that we will use two-stage least squares, and specify the model as follows:

```
LIST IVK C P(-1) E(-1) K(-1) G TIME TX W2;
2SLS(INST=IVK) CX C,W,P,P(-1);
FORM CONS ;
2SLS(INST=IVK) I C,P,P(-1),K(-1);
FORM INV ;
2SLS(INST=IVK) W1 C,E,E(-1),TIME;
FORM WAGES ;
IDENT WAGE W=W1+W2;
IDENT BALANCE P = CX+I+G - (TX+W) ;
IDENT PROFIT E = P+TX+W1 ;
IDENT CAPSTCK K=K(-1)+I ;

LIST KLEIN CONS INV WAGES WAGE BALANCE PROFIT CAPSTCK ;
LIST KENDOG CX I W1 W E P K ;
MODEL KLEIN,KENDOG,KLEINC ;
```

MODEL takes the model specified in the list of equations KLEIN and the list of endogenous variables KENDOG, and orders the equations into a recursive block structure. An ordering of this kind provides increased understanding of the model structure and provides efficient solution. The recursive block ordering for the system is formed by operations on the "adjacency" matrix of the system, a matrix of ones and zeroes relating the dependent variables in the system to the equations in which they appear.

See Steward (1962) for further explanation of this procedure and for details of the specific algorithm employed in obtaining the ordering. The method involves viewing the adjacency matrix as a network and systematically seeking closed loops that define systems of simultaneous equations.

The output for the MODEL example is shown in **Figure 14.1**. The column labeled Blk shows the number of each block of equations and the type (S for simultaneous and R for recursive). The X's mark the equations in which each dependent variable appears.

Because this model is completely simultaneous except for the capital stock equation, there is one block with six equations, followed by the equation that determines the new level of capital stock from investment. This last equation could be left out of the model without affecting the results in any way, since the level of the capital stock has no impact on output in this version of the model.

Having ordered this model, the procedure then stores the ordered list of equations and variables under the name KLEINC.

```
Block structure of KLEINC

Block    # Recursive    # Simultaneous
  #       equations        equations
  1                             6
  2            1

Blck Eq# Equation Dep.Var. 1234567
 1S    1 INV      I        X  X
 1S    2 WAGE     W         X   X
 1S    3 PROFIT   E          XX X
 1S    4 BALANCE  P        XX XX
 1S    5 CONS     CX        X XX
 1S    6 WAGES    W1         X  X
 2R    7 CAPSTCK  K        X     X
```

Figure 14.1 Sample MODEL Output for Klein Model I

Solution: SOLVE

SOLVE causes the model saved by a MODEL statement to be solved. The only required argument is the name of the ordered model. For each time period in the sample, each block is solved separately in the order determined by MODEL. The recursive blocks are always solved in one iteration via the equivalent of GENR. Three methods are provided for solving the simultaneous blocks: the Gauss-Seidel, Jacobi, and Fletcher-Powell algorithms.

Gauss-Seidel and Jacobi methods

Solution by the Gauss-Seidel method is the default, and it could be invoked for the Klein model in the following way (TAG is optional):

```
SOLVE(TAG=S) KLEINC;
```

The option METHOD=JACOBI specifies that a variant of the Gauss-Seidel method called the Jacobi method (see pp. 217-220 of Ortega and Rheinboldt) is to be used. This method does not update the endogenous variables immediately while the simultaneous block is being computed, but waits until the beginning of the next iteration and updates them all at once.

Fletcher-Powell method

To use the Fletcher-Powell method to solve the simultaneous blocks of a model, include the METHOD=FLPOW option in SOLVE:

```
SOLVE(METHOD=FLPOW) KLEINC ;
```

The Fletcher-Powell method is described in Fletcher and Powell (1963); it is useful when the Gauss-Seidel method does not converge. It uses numeric derivatives with respect to the endogenous variables (as opposed to SIML, which uses analytic derivatives).

Example: a 33-equation model

Figure 14.2 shows a sample TSP job for the solution of a rather complex 33-equation model with 6 structural equations and 27 identities. The first three pages show the TSP input for the model:

The data (10 observations in the example) are read from an external file, 6 variables per record, 14 records per observation. Some normalizing constants are defined.

The identities are specified; note that the equations are normalized, i.e., there is only one endogenous variable on the left-hand side of each equation, and each endogenous variable appears only once on the left-hand side. The six behavioral equations are defined, together with parameter values previously estimated by FIML over the whole sample. The endogenous variables and equation names are put into LISTs.

MODEL is executed to produce a collected model, TRADEC, to be input to SOLVE. We show the output of MODEL on the following page. This output shows the best order of solution of the model: the procedure has determined that the first 9 equations are recursive, the next 15 are a simultaneous block, and the last 9 are recursive once the previous 24 have been solved.

An inspection of the model confirms this: the first 7 equations involve only exogenous and lagged endogenous variables, equation IDFIO involves V and N, already determined by IDV and IDF9, equation IDRK involves RP, already determined by IDRP. Then we start the simultaneous block. Following this block is a series of equations that are essentially superfluous to the model: the whole thing could be solved without the last recursive section and then these equations could be computed using simple GENRs (in the order listed).

The following page shows the start of the solution procedure. Recursive blocks always take only one iteration, so no message is printed; the convergence messages shown correspond to the simultaneous blocks for the two periods. If we had used the PRINT option, there would also be a printout of SSR (sum of squared residuals) showing the golden section stepsize search method searching for the optional stepsize for each Fletcher-Powell iteration on the simultaneous block. Unlike the quasi-Newton minimization methods used elsewhere in TSP, the Fletcher-Powell algorithm does not have a natural stepsize of approximately unity.

Figure 14.2 Sample 33 equation trade model

```
NAME TRADESOL '33 EQUATION TRADE MODEL' ;
FREQ A ; SMPL 64 73 ;
READ (FILE='TRADESOL.DAT',FORMAT='(6G12.5)')
        ACDP AK AKD AKL CC CE
        CG DG DLDT DP DR E
        EI EJ EL ER ET EX
        G HLT HR I IG IM
        K KD L LD LE LG
        LH LJ LR LU M N
        NRE NW P PC PCE PCG
        PDP PEX PF PG PI PIG
        PIM PKD PL PLD PLE PLG
        PLR PR R RDP RE RIM RK
        RL RP RT RV RW SHRC
        SHRDP SHREX SHRFC SHRIM SHRKD T
        TDP TIM TK TL TP TV
        TW V VCR VIR W ;
?
```

```
?    SET THE CONSTANTS AND PARAMETERS FOR THE RUN.
?
SET NKD=166.92656 ;
SET NLD=252.81388 ;
SET NIM= 20.331 ;
SET NDP=419.56444 ;
SET NEX= 20.507 ;
SET  NT=  0.0 ;
SET  NL=  0.8464  ;
?
?    EQUATIONS OF THE 33 EQUATION MODEL.
?
IDENT IDSDP  DP=SHRDP*PLD*LD/PDP ;
? PROPER NORMALIZATION (BUT DOES NOT CONVERGE AS WELL):
?IDENT IDSIM PLD=-(PIM*IM)/(LD*SHRIM) ;
IDENT IDSIM  PL=PL+SHRIM+PIM*IM/(PLD*LD) ;
? PROPER NORMALIZATION:
?IDENT IDSKD  LD=-(PKD*KD)/(PLD*SHRKD) ;
IDENT IDSKD   L=L+SHRKD+PKD*KD/(PLD*LD) ;
IDENT IDSC   PC=(PL*LJ*SHRC)/(CC*(1-SHRC)) ;
IDENT IDSFC  CC=(SHRFC*(1+NW)*W(-1)-PL*LJ)/PC ;
IDENT IDF7  PDP=PC/ACDP ;
IDENT IDF8   LJ=LH-L ;
IDENT IDF9    N=(M*PI*AKL+PI(-1)*AK(-1)-PI*AKL
                -((1-TK)*((PKD*AKD)-TP*PI(-1)
            *AK(-1))))/(-PI(-1)*AK(-1)) ;
IDENT IDF10  NW=(-V-N*PI(-1)*AK(-1)*K(-1)
              + (PI*AKL-PI(-1)*AK(-1))*K(-1)
              -(1-TV)*(EI+VCR+VIR+NRE)
              -EJ+RV+TW*W(-1))/(-W(-1)) ;
IDENT IDF11 PEX=-(PDP*DP-PLD*LD-PKD*KD-PIM*IM)/EX ;
IDENT IDF12 IM = ((1+TDP)*PDP*DP-PC*CC+PCE*CE-PCG*CG
                   -PIG*IG-PI*I)/(-TIM*PIM)  ;
? DEPENDENT ON PROPER NORMALIZATION ABOVE:
?IDENT IDF13  PL=(PLD*LD+PLE*LE+PLG*LG+PLR*LR)*(1-TL)/L
;
IDENT IDF13 PLD=(PL*L/(1-TL)-PLE*LE-PLG*LG-PLR*LR)/LD ;
? DEPENDENT ON PROPER NORMALIZATION ABOVE:
?IDENT IDF14   L=LD+LE+LG+LR+LU ;
IDENT IDF14  LD=L-(LE+LG+LR+LU) ;
IDENT IDF16   R=(PEX*EX-PIM*IM+VCR+VIR+PLR*LR-ER-
HR+NRE-ET)/PR+R(-1) ;
IDENT IDDG DG=E+EL+ER+EI+EJ-
(RDP+RIM+RP+RK+RL+RW+RT+RE+RV);
IDENT IDDR DR=PEX*EX-PIM*IM+VCR+VIR-ER-HR+PLR*LR+NRE ;
IDENT IDE E=PCG*CG+PIG*IG+PLG*LG ;
IDENT IDG G=(DG+ET)/PG+G(-1) ;
IDENT IDKD KD=AKD*K(-1);
IDENT IDPF PF=PF(-1)*EXP(0.5*(
    ((PC*CC)/(PC*CC+PL*LJ)+(PC(-1)*CC(-1))/
    (PC(-1)*CC(-1)+PL(-1)*LJ(-1)))
   *LOG(PC/PC(-1))
   +((PL*LJ)/(PC*CC+PL*LJ)+(PL(-1)*LJ(-1))/
   (PC(-1)*CC(-1)+PL(-1)*LJ(-1)))
   *LOG(PL/PL(-1)))) ;
IDENT IDRDP RDP=TDP*PDP*DP ;
IDENT IDRE RE=PCE*CE-PLE*LE ;
IDENT IDRIM RIM=TIM*PIM*IM ;
IDENT IDRK RK=TK*(PKD*KD-RP)+TV*(EI+VCR+VIR+NRE);
IDENT IDRL RL=TL*(PLD*LD+PLE*LE+PLG*LG+PLR*LR);
IDENT IDRP RP=TP*PI(-1)*AK(-1)*K(-1);
IDENT IDRW RW=TW*W(-1);
IDENT IDV V=(PI*AKL-PI(-1)*AK(-1))*K(-1)
           +(PG-PG(-1))*G(-1)+(PR-PR(-1))*R(-1);
?
?     BEHAVIORAL EQUATIONS FOR THE CONSUMPTION SIDE OF
?  THE MODEL AND THEIR PARAMETER ESTIMATES.
?     SPECIFICATION: SYMMETRIC, CONVEXITY IMPOSED
?
FRML INTER1   SHRFC
     =((AX0+BON*LOG((PF*P)/
           ((1+NW)*W(-1)+LDA*(EL-HR-RT)+DEL*PL*LH))
           +BOT*(T-NT))/
     (-1+BNN*LOG((PF*P)/
           ((1+NW)*W(-1)+LDA*(EL-HR-RT)+DEL*PL*LH))
           +BNT*(T-NT)))*
     (1+(LDA*(EL-HR-RT)+DEL*PL*LH)/((1+NW)*W(-1)))$
FRML INTRA1 SHRC=AC+BCC*LOG(PC/(PL/NL)) ;
PARAM AX0 -.122007 BON -.0242706 LDA 7.85130
      DEL 7.85130  BNN -.197991 BNT .00143592
      BOT .000149361 ;
PARAM AC .148803 BCC .0155448 ;
?
?     BEHAVIORAL EQUATIONS FOR THE PRODUCTION SIDE OF
?   THE MODEL AND THEIR PARAMETERS.
?
FRML CODP SHRDP=ADP+(LDPKD*DKD
           -AKS*ADP)*LOG((KD*NEX)/(NKD*EX))
          +(LDPKD*LDPKD*DKD+LDPIM*LDPIM*DIM+DDP
          -ADP*(ADP-1))*LOG((DP*NEX)/(NDP*EX))
          +BDPT*(T-NT);
FRML COKD SHRKD=AKS
          +(DKD-AKS*(AKS-1))*LOG((KD*NEX)/(NKD*EX))
          +(LIMKD*DKD-AKS*AIM)*LOG((IM*NEX)/(NIM*EX))
          +(LDPKD*DKD-AKS*ADP)*LOG((DP*NEX) /(NDP*EX))
          +BKDT*(T-NT);
FRML COIM SHRIM=AIM+(LIMKD*DKD
          -AKS*AIM)*LOG((KD*NEX)/(NKD*EX))
          +(LIMKD*LIMKD*DKD+DIM
          -AIM*(AIM-1))*LOG((IM*NEX)/(NIM*EX))
          +(LDPKD*LIMKD*DKD
          +LDPIM*DIM-AIM*ADP)*LOG((DP*NEX)/
          (NDP*EX))+BIMT*(T-NT) ;
FRML COT DLDT=AT+BKDT*LOG((KD*NEX)/(NKD*EX))+
          BIMT*LOG((IM*NEX)/(NIM*EX))
          +BDPT*LOG((DP*NEX)/(NDP*EX))+BTT*(T-NT);
PARAM AT -.02217 BTT .0001963 ;
PARAM AKS -.6373 DKD 1.260 LIMKD .0881 AIM -.0794
      LDPKD -1.020 ADP 1.638 ;
PARAM BKDT -.0046 DIM -.0262 LDPIM -.4053 BIMT .0006
      DDP .0320 BDPT .0048 ;
?
?     THIS IS THE LIST OF ENDOGENOUS VARIABLES
?  IN THE MODEL.
?
LIST ENDOGL  E KD RE RP RW V N NW RK
             LJ LD PDP SHRC CC PF PLD
             SHRDP SHRKD SHRIM
             L PL DP SHRFC PC
             PEX R DLDT DR RDP RIM RL DG G ;
?
?      THIS IS THE LIST OF EQUATIONS IN THE MODEL;
?   THE ORDER CORRESPONDS
?   TO THE ORDER OF SOLUTION DESIRED.
?
LIST TRADEM IDE IDKD IDRE IDRP IDRW IDV IDF9 IDF10 IDRK
             IDF8 IDF14 IDF7 INTRA1 IDSFC IDPF IDF13

             CODP COKD COIM
             IDSKD IDSIM IDSDP INTER1 IDSC
             IDF11 IDF16 COT IDDR IDRDP
             IDRIM IDRL IDDG IDG ;
?
?      THIS COLLECTION OF THE MODEL IS SUPERFLUOUS
?  SINCE WE HAVE ALREADY ACHIEVED THE DESIRED ORDERING.
?
COLECT TRADEM ENDOGL TRADEC ;
SMPL 65 66 ;
SOLVE(STATIC,TAG=S,MAXIT=80) TRADEC ;  PRINT @IFCONV;
SOLVE(STATIC,METHOD=FLPOW,TAG=F,MAXIT=50,SQZTOL=.001)
TRADEC ;  PRINT @IFCONV;
?
?      SOLVE THE FIRST 2 BLOCKS (24 EQUATIONS) BY
? NEWTON'S METHOD.(WITH DEFAULT STARTING VALUES).
?
SIML (STATIC,ENDOG=(E KD RE RP RW V N NW RK
      LJ LD PDP SHRC CC PF PLD SHRDP SHRKD SHRIM
      L PL DP SHRFC PC))
      IDE IDKD IDRE IDRP IDRW IDV IDF9 IDF10 IDRK
      IDF8 IDF14 IDF7 INTRA1 IDSFC IDPF IDF13
      CODP COKD COIM
      IDSKD IDSIM IDSDP INTER1 IDSC ;
PRINT @IFCONV;
?
?      SOLVE THE FIRST 2 BLOCKS (24 EQUATIONS) BY
? NEWTON'S METHOD. (WITH STARTING VALUES FROM
?GAUSS-SEIDEL).
?
DOT E KD RE RP RW V N NW RK
    LJ LD PDP SHRC CC PF PLD SHRDP SHRKD SHRIM
    L PL DP SHRFC PC;
  .SAV = .;
  . = .S;   ? USE SOLVED VALUES AS STARTING VALUES
ENDDOT;
SIML (STATIC,TAG=N,ENDOG=(E KD RE RP RW V N NW RK
      LJ LD PDP SHRC CC PF PLD SHRDP SHRKD SHRIM
      L PL DP SHRFC PC))
     IDE IDKD IDRE IDRP IDRW IDV IDF9 IDF10 IDRK
     IDF8 IDF14 IDF7 INTRA1 IDSFC IDPF IDF13
     CODP COKD COIM
     IDSKD IDSIM IDSDP INTER1 IDSC ;
PRINT @IFCONV;
?
?     PRINT THE RESULTS FROM ALL THREE METHODS.
?
TITLE 'COMPARE RESULTS FROM GAUSS, FLPOW, AND NEWTONS
METHOD' ;
PAGE ;
DOT E KD RE RP RW V N NW RK
    LJ LD PDP SHRC CC PF PLD SHRDP SHRKD SHRIM
    L PL DP SHRFC PC;
  . = .SAV;   ? RESTORE ORIGINAL VARIABLES
  PRINT .N .S .F . ;
ENDDOT;
```

```
Block structure of TRADEC

Block    # Recursive    # Simultaneous
  #       equations        equations
  1            9
  2                            15
  3            9

                                    111111111122222222223333
Blck Eq# Equation Dep.Var. 123456789012345678901234567890123
 1R    1 IDE      E        X
 1R    2 IDKD     KD        X
 1R    3 IDRE     RE         X
 1R    4 IDRP     RP          X
 1R    5 IDRW     RW           X
 1R    6 IDV      V             X
 1R    7 IDF9     N              X
 1R    8 IDF10    NW            XXX
 1R    9 IDRK     RK        X X    X
 2S   10 IDF8     LJ                X         X
 2S   11 IDF14    LD                 X        X
 2S   12 IDF7     PDP                 X           X
 2S   13 INTRA1   SHRC                 X       X  X
 2S   14 IDSFC    CC              X X   X      X XX
 2S   15 IDPF     PF                X   XX     X  X
 2S   16 IDF13    PLD                655X    X   XX
 2S   17 CODP     SHRDP     X              X    X
 2S   18 COKD     SHRKD     X               X   X
 2S   19 COIM     SHRIM     X                X  X
 2S   20 IDSKD    L         X        X    X X X
 2S   21 IDSIM    PL                 X    X  X X
 2S   22 IDSDP    DP                 XX   XX    X
 2S   23 INTER1   SHRFC           X      X     X X
 2S   24 IDSC     PC                X  XX      X  X
 3R   25 IDF11    PEX       X        XX   X     X  X
 3R   26 IDF16    R                                 XX
 3R   27 COT      DLDT      X                   X    X
 3R   28 IDDR     DR                                X  X
 3R   29 IDRDP    RDP                 X         X      X
 3R   30 IDRIM    RIM                                   X
 3R   31 IDRL     RL                 X    X              X
 3R   32 IDDG     DG       X XXX   X                   XXXX
 3R   33 IDG      G                                       XX

Current sample:  1965 to 1966

                        SIMULATION OF THE MODEL TRADEC
                        ==============================

       METHOD = Gauss-Seidel

Number of Equations in the model =  33
Number of Blocks in the model = 3

         Block #   Number of Equations
               1          9
               2         15
               3          9
ITERATION LIMIT IN BLOCK      2
LAST SUM OF SQUARED RESIDUALS =  0.10432128
PROGRAM CONTINUING USING LAST VALUES COMPUTED.
ITERATION LIMIT IN BLOCK      2
```

```
LAST SUM OF SQUARED RESIDUALS =  0.24700841E-01
PROGRAM CONTINUING USING LAST VALUES COMPUTED.

  320 FUNCTION EVALUATIONS.

THE SOLVED VARIABLES  ARE STORED WITH A TAG:    S
                              SIMULATION RESULTS
....printout of results

               @IFCONV
1965           1.00000
1966           1.00000

                        SIMULATION OF THE MODEL TRADEC
                        ==============================

       METHOD = Fletcher-Powell

Number of Equations in the model =  33
Number of Blocks in the model = 3

         Block #   Number of Equations
               1          9
               2         15
               3          9

Working space used: 4129
                                STARTING VALUES
......
F= .4655745E-03  FNEW= .4628242E-03  ISQZ=  1 STEP= 1.       CRIT=
.29951E-05

CONVERGENCE ACHIEVED AFTER  13 ITERATIONS
 1020 FUNCTION EVALUATIONS.

F= .8875898E-03  FNEW= .8853670E-03  ISQZ=  1 STEP= 1.       CRIT=
.22314E-05

CONVERGENCE ACHIEVED AFTER  38 ITERATIONS
 4138 FUNCTION EVALUATIONS.

THE SOLVED VARIABLES  ARE STORED WITH A TAG:    F
                              SIMULATION RESULTS
.....printout results

               @IFCONV
1965           1.00000
1966           1.00000

         SOLVE THE MODEL BY NEWTONS METHOD USING DEFAULT STARTING VAL
         ============================================================

....... 1965 did not converge......
CONVERGENCE NOT ACHIEVED AFTER  20 ITERATIONS

                                 PERIOD: 1965
               @IFCONV
1965           0.00000
1966           1.00000

         SOLVE WITH NEWTONS METHOD USING GAUSS-SEIDEL STARTING VALUES
         ============================================================
                               MODEL SIMULATION
                               ================

                 STATIC SIMULATION

Working space used: 9487
                                STARTING VALUES
.....
F= .6452811E-12  FNEW= .1875796E-24  ISQZ=  0 STEP= 1.       CRIT=
.64528E-12

CONVERGENCE ACHIEVED AFTER   3 ITERATIONS
    6 FUNCTION EVALUATIONS.

F= .5294763E-14  FNEW= .3275911E-25  ISQZ=  0 STEP= 1.       CRIT=
.52948E-14

CONVERGENCE ACHIEVED AFTER   3 ITERATIONS
   12 FUNCTION EVALUATIONS.

THE SOLVED VARIABLES  ARE STORED WITH A TAG:    N
                              SIMULATION RESULTS
......output omitted

               @IFCONV
1965           1.00000
1966           1.00000

             COMPARE RESULTS FROM GAUSS, FLPOW, AND NEWTONS METHOD
             =====================================================

                    EN            ES            EF             E
1965         137.00119     137.00119     137.00119     137.00999
1966         156.81017     156.81017     156.81017     156.81000

......output omitted

                    GN            GS            GF             G
1965         316.92780     317.54471     317.71494     317.38000
1966         327.72321     327.86340     323.81900     327.85001

            Compute correlation across solutions with actual values
            =======================================================

Current sample:  1965 to 1965
Current sample:  1965 to 1965
Current sample:  1 to 33

                        Results of Covariance procedure
                        ===============================
Number of Observations: 33
                              Correlation Matrix

                  SOLN          SOLS          SOLF        ACTUAL
SOLN           1.00000
SOLS           0.99998       1.00000
SOLF           0.99946       0.99964       1.00000
ACTUAL         0.99949       0.99966       0.99999        1.0000
```

Figure 14.3 33 Equation Trade Model Output

Displaying and evaluating a forecast: ACTFIT

The first step in evaluating a forecast is probably to print or plot it. If you are using the single equation FORCST, this is easily accomplished by including the PRINT option on the statement:

```
FORCST(PRINT) SALESF ;
```

This option prints information about the equation it is using for the forecast and a time series plot of the forecasted variable over the time period in question. The plot also shows the values of the variable on the right-hand side. On a PC, to see a high-resolution graphics plot use the command :

```
PLOT SALES SALESF ;
```

If you use SOLVE or SIML to obtain the forecast, plotting will not be automatically available, although you can obtain a table of the solved values by use of the PRNSIM option. To get plots, save the simulated values with a TAG= option (for example, CXS, IS, etc.) and use the PLOT procedure described in Chapter 6:

```
PLOT CX A CXS S ;     ? printer format
PLOT CX CXS ;         ? high-resolution graphics
```

This example assumes that the simulation has been done over a historical period, so that both actual (CX) and solved (CXS) values of consumption are available and may be plotted on the same scale for comparison. Of course, if you did not know CX for the forecasting period, you could simply plot CXS.

Standard references for the evaluation of forecasts are: Theil (1961 and 1966) and Pindyck and Rubinfeld (1976). Some of the measures discussed by Theil in Chapter 2 of his 1966 book have been incorporated into TSP in the ACTFIT procedure. You can use this procedure to compute a set of statistics such as the root mean square error, the inequality coefficient (*U*), and a decomposition of the sources of forecast error. The command is

```
ACTFIT CX CXS ;
```

Once again, the historical values of the variable are required to make this comparison. If the option PLOTS has been turned on, ACTFIT also plots the two variables and their difference.

Note: Theil's 1961 and 1966 definitions of *U* differ; Pindyck and Rubinfeld use the 1961 definition. TSP prints both versions of *U*.

14.5 Monte Carlo Simulation: RANDOM

Random variables are useful in a variety of situations, from checking the statistical properties of an econometric or simulation model to computing bootstrap standard errors when analytic formulas are not available. This type of procedure is often referred to as Monte Carlo analysis, and is easily programmed in TSP. For example, to draw a series E of independent standard normal variates, use the following command:

```
RANDOM E ;
```

To draw 3 series X1, X2, X3 that have a multivariate normal distribution with mean XMEAN (a length 3 vector) and variance matrix XVAR (a symmetric matrix of order 3), use this command:

```
RANDOM (VMEAN=XMEAN,VCOV=XVAR) X1-X3 ;
```

Random can also be used to generate series from uniform, Poisson, Cauchy, exponential, Laplace, student's t, gamma, negative binomial, and arbitrary empirical distributions. See the *Reference Manual* for details on the options necessary.

Two useful examples appear in Chapter 9: in section 9.6.3, we simulate an ARCH model using the standard normal distribution and a dynamic GENR. In section 9.6.7, we show how to use uniform random variables and the inverse distribution function to generate random variates from an arbitrary distribution function, in this case, the Type I Extreme Value Distribution [exp(-exp(-u))].

The example below uses the random number generator to make a chi-squared(3) random variable and then plots the "empirical" distribution of this random variable on the same scale as the theoretical distribution (computed by CDF). The plot is shown in **Figure 14.4**.

```
SMPL 1 100 ;
RANDOM E1-E3 ;       ? Make 3 indep. normal RVs.
```

```
?    Chi-squared variable=sum of squares of normal RVs.
CHI3 = E1*E1+E2*E2+E3*E3 ;

? These statements make the empirical distribution function of chi3
SORT CHI3 ;
TREND T ;
P = T/@NOB ;            ? p = 1/n,2/n,etc. is the height at the
          corresponding value chi3.
SET MAXC = 1+INT(1.1*CHI3(@NOB));       ? maxc is the upper
          limit of the graph.

? Make the theoretical chi-squared(3) distribution function using CDF
CHIVAL = MAXC*(T-1)/(@NOB-1) ;
CDF(CHISQ,DF=3,LOWTAIL) CHIVAL CHIDF3 ;

GRAPH(PAIR,LINE,TITLE="Reference vs Actual Distribution")
          CHIVAL,CHIDF3 CHI3,P ;
```

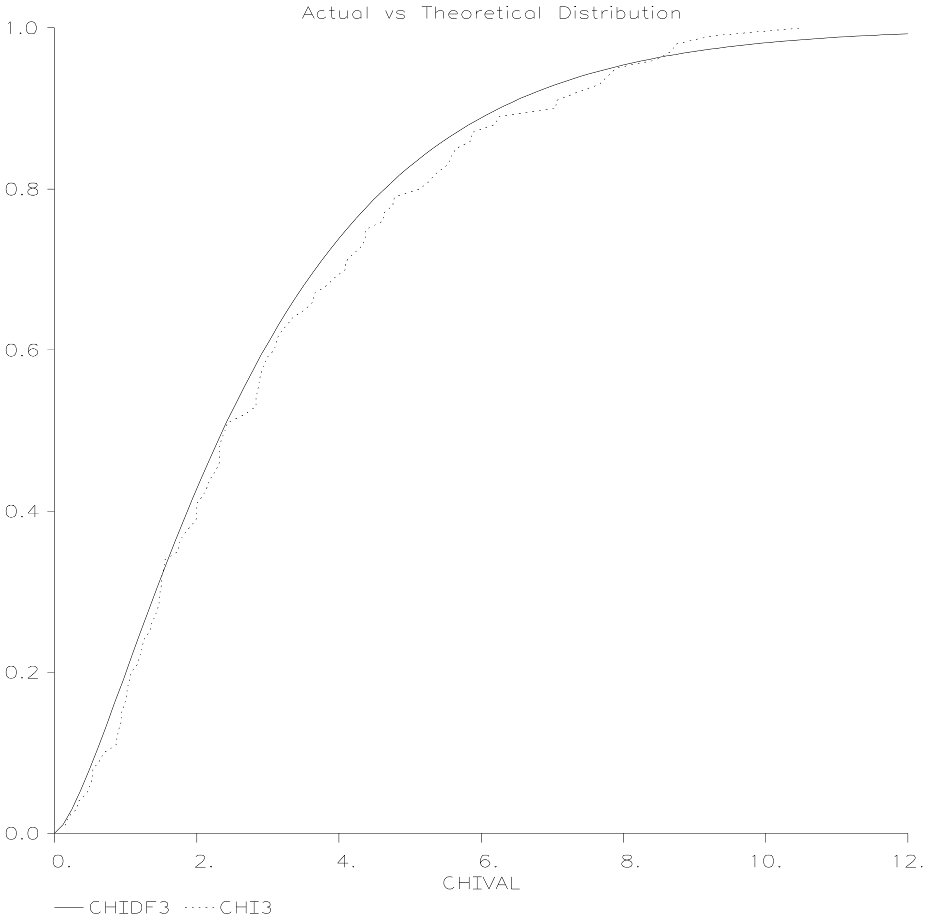


Figure 14.4 Simulating the chi-squared distribution

Most Monte Carlo analysis involves making a large loop and accumulating statistics on functions of random variables. As an illustration of how to do this in TSP, the distribution of the OLS regression coefficients can be checked under the standard assumptions of fixed Xs and normal residuals.

```
SET NTRIAL=100; SET NOB=50; SET NX=2;
SMPL 1,NOB;

? generate X series; use SEEDIN to make results reproducible.
RANDOM(MEAN=5,SEEDIN=49824) X;
YHAT = 3 + 4*X;                 ? generate true values of Y series
```

```
MFORM(NROW=NX,NCOL=1) BMEAN=0;
MFORM(NROW=NX,TYPE=SYM) BPROD=0;

? suppress all OLS output.
REGOPT(NOPRINT) @LOGL,@COEF,@SES;
DO TRIAL = 1,NTRIAL;
        RANDOM(STDEV=2) E;         ? generate disturbances E
        Y = YHAT+E;
        OLSQ(SILENT) Y C X;
        MAT BMEAN = @COEF+BMEAN ;      ? sum coeff. estimates
        MAT BPROD = BPROD+@COEF*@COEF' ; ? cross products
ENDDO;
MAT BMEAN = BMEAN/NTRIAL ;
MAT BVAR = BPROD/NTRIAL - BMEAN*BMEAN' ;
REGOPT;                            ? reset output suppression.
TSTATS(NAMES=@RNMS) BMEAN BVAR;
```

For lengthy Monte Carlo studies, it may be useful to write intermediate results occasionally to a file, so that if the program fails for some reason partial results can be obtained. This would also yield more accurate estimates of the variance (the updating formula used above is not very accurate). That is, revise the example above as follows:

```
 ...
DO TRIAL = 1,NTRIAL;
        RANDOM(STDEV=2) E;
        Y = YHAT+E;
        OLSQ(SILENT) Y C X;
        WRITE(FILE='monte1.dat') @COEF;
ENDDO;

? The second part of the program (below) could be put in a separate
        file if necessary
SMPL 1 NTRIAL;
READ (FILE='monte1.dat') B0 B1;
MSD (COVA) B0 B1 ;
REGOPT;                      ? reset output suppression.
? Display mean and variance of estimated coefficients
TSTATS(NAMES=(C,X)) @MEAN @COVA;
```

15. PANEL DATA

The term "panel data" usually refers to data where the unit of observation varies in two or more dimensions. For example, you might have a sample of the same individuals observed at several points in time, or a set of time series, each for a different firm or country. The analysis of panel data in economics has become increasingly important in recent years as the number of such datasets has grown along with econometric techniques to analyze them. These data can be handled rather easily in TSP, although the inherent complexity of the data structure requires you to think a little harder about how to set things up.

This chapter provides some guidance on how to analyze panel data in TSP.. We begin with a few basic rules for setting up your data input, depending on the nature of the problem you are analyzing. Then we discuss the PANEL command, which produces total (pooled), between, within (fixed effects or conditional), and variance components (random effects) estimates for panel data. This is followed by a discussion of how to use panel data in nonlinear models such as Probit. Finally we discuss how to estimate more complicated models in short panels using the minimum distance estimator in LSQ or GMM. This is a powerful methodology that can be used to estimate linear, nonlinear, and dynamic factor models with panel data, using the method of moments estimator to obtain asymptotically efficient estimates.

In the first part of this chapter we use the example of data on the patenting and R&D spending of a large number of firms, for three years each (see Hall, Griliches, and Hausman 1986; we use three years of data for simplicity, more would be needed to obtain real answers). Although the underlying patent data are application counts, we confine the analysis here to large firms so we can treat the patents as a continuous rather than discrete variable.

In the example, we are interested in the relationship between R&D spending (possibly lagged) and the resulting patent applications. The specification of the model that seems to have the most stable properties is to regress the log of patents on contemporaneous and lagged logs of R&D expenditures. However, as will become clear in the example, we expect a fixed difference in the propensity to patent across firms (because of differing technological characteristics of the industry and other reasons), and we expect that this propensity may be correlated with the level of R&D expenditure. This leads us to use many of the panel data techniques described in this chapter.

15.1 The basics of using panel data

15.1.1 Reading in panel data

The first decision to make when dealing with panel (time series-cross section) data is how to organize it. Usually, you are willing to assume conditional independence in one direction, but not in the other. For example, you may be willing to assume that observations on firms are conditionally, independently, and identically distributed, but not that there is no serial correlation within a set of observations on a single firm. If this is the case you should order the data so that the slowest varying index is the index of the dimension in which the data are independent. In the example of patents and R&D, where we have data for N firms for 3 years (1974, 1975, 1976), use the order

FIRM	TIME PERIOD	VARIABLE
1	74	Patents, R&D for firm 1 in year 74
1	75	Patents, R&D for firm 1 in year 75
1	76	Patents, R&D for firm 1 in year 76
2	74	Patents, R&D for firm 2 in year 74
.	.	.
N	74	Patents, R&D for firm N in year 74
N	75	Patents, R&D for firm N in year 75
N	76	Patents, R&D for firm N in year 76

This order facilitates the construction of estimators that include serial correlation. It also allows you to regroup the datafile easily so that there is one cross section unit per observation with all the variables for all years (by reading in a different format), for use with the robust short panel methods of section 3.1. For example,

FIRM	VARIABLES
1	Patents, R&D for year 74; Patents, R&D for year 75;...
2	Patents, R&D for year 74; Patents, R&D for year 75;...
.	.
N	Patents, R&D for year 74; Patents, R&D for year 75;...

Here are two READ commands that read the same dataset into TSP in the two different formats shown above. The first method is the following:

```
SET NOBS = 3*N ;
```

```
SMPL 1 NOBS ;
FREQ (PANEL, T=3, ID=@ID) ;
READ (FILE='PATDATA.DAT') @ID PATENTS LRND ;
```

Compare to the second method, which reads all the data for a firm into a single observation:

```
SMPL 1 N ;
READ (FILE='PATDATA.DAT') @ID PAT74 LRND74 @ID PAT75
        LRND75 @ID PAT76 LRND76 ;
? note that only the value of @ID for 1976 will be stored.
```

There are situations where you want the data one observation per firm-year (the PANEL command, Probit or AR1 with the FE or RE options) and situations where you want the data one observation per firm (when using complex lag structures or GMM methods); thus the data should be set up with this in mind. We refer to the first format as the *pooled* format and the second format as the *panel* format.

Several factors will influence your choice between these two formats:

1. The statistical assumption of conditional independence: in general, when you assume that observations are independent (conditional on your model) across both dimensions, you will want the data in pooled format, and when you assume independence only in one direction (with the possible exception of simple first order serial correlation), the panel format.
2. Whether the data are unbalanced (see the next section). If the unbalancing is severe, the pooled format is far more efficient because you will not have to create variables for the many time periods that are missing.
3. Whether the model you have in mind has parameters that are constant across all the periods. Such models are easily specified using data stored in the pooled format. When the parameters vary across time periods, the panel structure may be easier to work with.

15.1.2 Unbalanced panels

An unbalanced panel is one where there are a different number of observations for each cross section unit (or vice versa). These observations may be contiguous, or there may be holes in the data. That is, for the example dataset, we may have four years of data for one firm (1973 to 1976), three years for another (1973, 1975, 1976) and two years for a third (1975 and 1976). It is essential to use the FREQ(PANEL,ID=*ID_variable*); command with an unbalanced panel, to identify an *ID_variable* that indicates when one firm stops and the next one starts.

Many, but not all, of the estimators described in this chapter can be used with unbalanced panels. For example, the PANEL procedure, which assumes independence across time series and cross section units will work just the same whether the data are balanced or unbalanced. In fact, if your data are unbalanced, PANEL will tell you so and will display the Ahrens-Pincus unbalancedness measure (*APUI*). This measure is a number between zero and one, with values closer to zero indicating more unbalanced data. *APUI=1* means the data is balanced.

All command which use lags and leads recognize the *ID variable*, so that lags and leads will refer only within a single individual. One implication of this is that you can use a command like

```
SELECT MISS(IDV(-1));        ? where IDV is the ID variable
```

to choose the first observation for all individuals. The AR1 command also recognizes the *ID variable*, so it will apply the special transformation to the first observation of each individual and avoid using any lags that would refer from one individual back to the previous individual. When the PLOT procedure encounters panel data, it will place a break in the line to indicate the start of data for a new individual.

In the panel format, unbalanced panels can be "balanced" by including missing data codes for the missing observations. Some of the methods described in section 15.4.1 may not work very well with unbalanced panels. This is in the nature of the data and the current state of econometric methodology; it is not necessarily a limitation of the program.

15.2 Random and Fixed Effects models -- the PANEL procedure

PANEL obtains estimates of linear regression models for panel data (several observations or time periods for each individual). The data may be unbalanced (different number of observations per individual). PANEL can also compute means by group and perform F tests between groups. To define the models estimated, assume we have observations on $i=1,...,N$ individuals for each of $t=1,...,T$ years. The dependent variable is denoted by y_{it} and the independent variables by X_{it}. The basic pooled or TOTAL regression model is

$$y_{it} = X_{it}\beta + \alpha_0 + u_{it}$$

where α_0 is the overall intercept and u_{it} assumed to be i.i.d. This model assumes a single set of slope coefficients for all the observations.

The fixed effect or WITHIN model assumes that there are common slopes, but that each cross section unit has its own intercept, which may or may not be correlated with the Xs:

$$y_{it} = X_{it}\beta + \alpha_i + u_{it}$$

The BYID model assumes that both the slopes and the intercepts vary across cross section units:

$$y_{it} = X_{it}\beta_i + \alpha_i + u_{it}$$

The BETWEEN model specifies the same relationship between the individual means:

$$y_{i\bullet} = X_{i\bullet}\beta + \alpha + u_{i\bullet}$$

where

$$y_{i\bullet} = \frac{1}{T_i}\sum_{t=1}^{T_i} y_{it}$$

The random effects or VARCOMP model resembles the WITHIN model, but it assumes that the intercepts are drawn from a common distribution with mean α and variance σ_α^2. Unlike the WITHIN model, the estimates for this model will not be consistent if the individual intercepts are correlated with the independent variables. Because of this, it is important to test for correlation. PANEL reports the Hausman test statistic for the difference between the fixed effects and random effects estimates, along with its p-value.

The VARCOMP estimator is computed by estimating the relative importance of between and within variation of the disturbance $\alpha_i + u_{it}$ and using this estimated ratio to combine the within and between estimators optimally. Under the null of uncorrelated intercepts, the VARCOMP estimator is asymptotically efficient, since it is a generalized least squares estimator. There are additional options for VARCOMP to control the actual variance components. Small or large sample formulas may be used, or you can supply the values directly. If negative variances are computed using the small sample formula, the program switches over to the large sample formulas, which always result in positive values.

All or some of these models can be estimated by a single PANEL statement. The basic PANEL statement is like the OLSQ statement: first list the dependent variable and then the independent variables. C is optional; an intercept term is essential for these models and will be added if not present. Here is an example for the sample dataset:

```
PANEL LPAT C LRND ;
```

This command will produce estimates of the TOTAL, WITHIN, BETWEEN, and VARCOMP models, together with the value of an F-statistic for the hypothesis that all the intercepts are equal. The observations over which the models are computed are determined by the current sample. Lags and leads are not allowed to extend from one individual's data to another's data.

Your data must be set up with all the time periods for each individual together (in pooled format). You must also specify when the data ends for one individual, and begins for the next. The best method is to provide an *ID variable* series in the FREQ(PANEL) command that takes on different values for each individual, as we did in the sample dataset. If your data are balanced (the same number of time periods for every individual), the T= option can be used. Other options are also available (see the *Reference Manual* or *Online Help System*). If the data are not in this order, the SORT command can reorder them (you can also use SORT to reorder the data so that you can do variance components in the other (time) dimension). See Section 6.4 for an example.

Frequently you will also wish to include a set of dummies for the time periods in your estimation, especially when your panel is short (so that there is very little sacrifice in terms of degrees of freedom). To generate a set of time dummies for the sample dataset, use the TREND and DUMMY commands:

```
?   Make a series = 74,75,76,74,75,76,... (for balanced data)
TREND (PER=3,START=74) YEAR ;
LIST YRDUM YEAR74-YEAR76 ;
DUMMY YEAR YRDUM ;
```

This creates three variables YEAR74, YEAR75, and YEAR76 with the following values:

```
 OBS   YEAR74        YEAR75        YEAR76
 1,1   1             0             0
 1,2   0             1             0
 1,3   0             0             1
 2,1   1             0             0
 2,2   0             1             0
…..and so forth
```

If you had loaded a variable YEAR (which would have been essential if your data had been unbalanced), you could have just used the DUMMY command directly, without using TREND.

The next example estimates all models including the individual firm regressions, and prints individual means:

```
PANEL(MEAN,BYID) LPAT C LRND YEAR75 YEAR76 ;
```

The output for this command will include F-statistics for the hypothesis that the slope coefficients are equal and for the joint hypothesis that both the slopes and intercepts are equal.

The following command estimates VARCOMP only, using large sample formulas (note the use of year dummies with the intercept):

```
PANEL(NOTOT,NOBET,NOWITH,NOVSMALL) LPAT LRND C YEAR75
        YEAR76;
```

The TSP International website contains an example that shows all the linear model Panel estimators available, estimated using Grunfeld's investment data and compared to the published results of Baltagi (1995) and Nerlove (2000).

15.3 Robust estimation with panel data

This section discusses how to obtain asymptotically efficient estimates of panel data models without imposing conditional homoskedasticity or independence over time on the disturbances of the model. The first two methods we describe (Chamberlain's PI matrix method and the estimation of a dynamic factor model using variance components) are based on the idea that the second and fourth moments of the data are sufficient statistics for estimation of a linear model with heteroskedastic disturbances. The final estimator we discuss is the well-known GMM method for panel data originally proposed by Hansen and Singleton (1982) and made popular for panel data by Arellano and Bond (1991).

The estimators described here are minimum distance estimators that use an asymptotically optimal weighting matrix. In particular, they use the sample covariance of the distance measures (for example, the orthogonality conditions in GMM) as the weighting matrix. The key to understanding the computation of the first two estimators described here is to recognize that the SUR procedure (which computes multivariate regressions without imposing a diagonal covariance structure) can also compute a set of estimates of second moments or functions of second moments, *along with a robust estimate of their variance-covariance matrix*. This estimate is heteroskedastic-consistent and does not impose independence across the disturbances in each equation, where equation here refers to the moment equation.

This enables you to construct the optimal weighting matrix for many of these estimators easily, without special programming. Using this matrix, which we call OMEGA, we can construct a minimum distance estimator for the second moments as functions of the parameters of interest using the SUR procedure again, but this time with only one observation (since the second moments and the estimated OMEGA are sufficient statistics for the problem).

This methodology is applied below to two different panel data estimation problems: the problem of describing the relationship between a set of endogenous variables (Y) and a set of exogenous variables (X), where the reduced form Π matrix is a sufficient statistic for the problem (Chamberlain's problem), and the problem of describing the relationship between a set of endogenous variables (Y) and a set of unobservable variables ("factors"), where the second moments of Y are a sufficient statistic. Obviously, the two types of models could be combined, but the presentation is simpler if they are treated separately.

Section 15.3.4 then describes how to use the GMM procedure in TSP to estimate linear and nonlinear panel data models that allow for both heteroskedasticity and autocorrelation of the disturbances.

15.3.1 Obtaining panel-robust standard errors

Because estimates based on linear models and some nonlinear models (notably those estimated by ML using a likelihood function of the exponential form) are often consistent even in the presence of heteroskedasticity or a more complex variance structure, sometimes you will simply wish to compute robust standard errors for conventional estimates rather than using one of the estimators below.

TSP provides several such options. You can use the ROBUST option in the linear estimation commands or HCOV=W in the nonlinear estimation commands to obtain heteroskedastic-consistent standard errors. In this case, if FREQ(PANEL) has been specified, OLSQ, PANEL, and 2SLS will compute standard errors that are robust to both heteroskedasticity and autocorrelation within panel unit. You can override the autocorrelation computation by using the combination of options ROBUST, HCOMEGA=DIAGONAL.

In PROBIT (REI or FEI) the same option is implemented using HCOV=P (block diagonal variance-covariance matrix robust to autocorrelation but not heteroskedasticity) and HCOV=Q (robust to both autocorrelation and heteroskedasticity). Future versions of TSP will implement these options in other nonlinear estimation procedures.

15.3.2 The PI matrix method

Chamberlain (1982) showed that one way to estimate a whole range of panel data models was to summarize the data by regressing all the endogenous variables on all of the exogenous variables, obtaining an estimate of the reduced form matrix *Π*; and then to test various restrictions on this matrix implied by the models of interest (actually Chamberlain focused on the conditional expectation interpretation of regression so that the estimator in question was for the expectation of the *Y*s conditional on the *X*s). Let π = *vecΠ*. Then, if you use the minimum distance estimator

$$\arg\min_{\delta} \; (\pi - f(\delta))\Omega^{-1}(\pi - f(\delta))$$

together with an appropriate estimate of Ω to estimate the restricted parameter set δ, then the resulting estimates of δ are asymptotically efficient. The optimal estimate of Ω in this case is given by the sample covariance of w_i, where

$$w_i = (y_i - \Pi x_i) \otimes S_x^{-1} x_i$$

and S_x is the sample variance of the *X*s. Note that this formula does not imply independence *within* each observational unit, nor does it impose homoskedasticity.

Using SUR, it is easy to estimate Π and its associated covariance Ω in TSP. For the sample dataset:

```
DOT 74-76 ;
FRML PIEQ. LPAT. = PI.74*LRND74 + PI.75*LRND75 +
        PI.76*LRND76 ;
PARAM PI.74-PI.76 ;
MSD (NOPRINT) LPAT. LRND. ; ? Removing all the year means.
UNMAKE @MEAN PMEAN RMEAN ;
LPAT. = LPAT.-PMEAN ; LRND. = LRND.-RMEAN ;
ENDDOT ;
SUR (HCOV=R) PIEQS ;         ? Note the robust option.
COPY @COEF PI ;              ? Save the computed PI matrix
COPY @VCOV OMEGA ;           ? and its covariance estimate.
```

Removing the means of the data before forming the estimated Π simplifies things, because it implies that you do not have to carry around the X variable corresponding to the intercept. This two part strategy for estimation does not affect the asymptotics (Macurdy 1982).

Now suppose the class of restricted models of interest have the following form:

$$y_{it} = \beta_1 x_{it} + \beta_2 x_{i,t-1} + \ldots + \gamma_t \alpha_i + u_{it}$$

where α_i is the firm effect, which may be correlated with the *x*'s:

$$\alpha_i = \lambda_1 x_{i1} + \ldots + \lambda_T x_{iT}$$

For the sample data, with three *y*'s and three *x*'s, the Π matrix has the following form:

$$\Pi = \begin{bmatrix} \beta_1 + \gamma_{74}\lambda_{74} & \gamma_{74}\lambda_{75} & \gamma_{74}\lambda_{76} \\ \beta_2 + \gamma_{75}\lambda_{74} & \beta_1 + \gamma_{75}\lambda_{75} & \gamma_{75}\lambda_{76} \\ \beta_3 + \gamma_{76}\lambda_{74} & \beta_2 + \gamma_{76}\lambda_{75} & \beta_1 + \gamma_{76}\lambda_{76} \end{bmatrix}$$

There are nine elements in Π and nine coefficients to be estimated, but there is one normalization restriction ($\gamma_{74}=1$), so there is one over-identifying restriction. If there are no correlated effects, the γs and λs will be zero and there will be six over-identifying restrictions.

With the estimated Π and Ω matrices obtained above, you can test for the two levels of restrictions implied by this model:

1) a stable lag structure and correlated firm effect.
2) a stable lag structure and uncorrelated firm effects.

Here is how to do it using the minimum distance procedure (LSQ or SUR):

```
? Define the lists of PI coefficients and equations.
?
LIST PILIST PI7474-PI7476 PI7574-PI7576 PI7674-PI7676 ;
LIST PIEQLIST PIEQ7474-PIEQ7476 PIEQ7574-PIEQ7576
        PIEQ7674-PIEQ7676 ;
LENGTH PILIST NPI ; ? Find out how many elements in PI matrix.
? Unmake the est. PI matrix into its individual elements.
UNMAKE PI PILIST ;
? Treat the estimated PI coefficients as data in the program below
CONST PILIST ;
? Suppress some output for Minimum Distance estimation:
SUPRES COVU W COVT REGOUT;
? Define the equations that express PI as a function of the
        underlying delta parameters ? (beta, lambda, and gamma)
        that are to be estimated.
?
FRML PIEQ7474 PI7474 = BETA1 + LAM74*GAM74 ;
FRML PIEQ7574 PI7574 = BETA2 + LAM74*GAM75 ;
FRML PIEQ7674 PI7674 = BETA3 + LAM74*GAM76 ;
FRML PIEQ7475 PI7475 = LAM75*GAM74 ;
FRML PIEQ7575 PI7575 = BETA1 + LAM75*GAM75 ;
FRML PIEQ7675 PI7675 = BETA2 + LAM75*GAM76 ;
FRML PIEQ7476 PI7476 = LAM76*GAM74 ;
FRML PIEQ7576 PI7576 = LAM76*GAM75 ;
FRML PIEQ7676 PI7676 = BETA1 + LAM76*GAM76 ;

? Starting values for model with
? uncorrelated Xs (only betas to be estimated).
CONST LAM74-LAM76 GAM74-GAM76 ;
PARAM BETA1 1 BETA2 0 BETA3 0 ;
CONST LAM74 1 ;    ? this is a free normalization

? In effect we now have one observation on each element of PI.
SMPL 1,1 ;
SUR(WNAME=OMEGA) PIEQLIST ;
LENGTH @RNMS NOPAR ;
SET DF = NPI-NOPAR ;  ? Degrees of freedom for constrained model
CDF(CHISQ,DF=DF) @TR ;              ? Test constraints.

? Starting values for model with correlated Xs.
PARAM LAM74-LAM76 GAM75 GAM76 ;
 SUR(WNAME=OMEGA) PIEQLIST ;
LENGTH @RNMS NOPAR ;
SET DF = NPI-NOPAR ;
CDF(CHISQ,DF=DF) @TR ;    ? Test the single constraint remaining.
```

The "TRACE OF MATRIX" criterion printed out by SUR after convergence is precisely the Chi-squared statistic for the over-identifying constraints (with degrees of freedom equal to the number of elements of Π less the number of parameters being estimated). Note that changing the sample size to one is essential if you want standard error estimates to have the correct size (assuming OMEGA has been computed as shown).

With a larger number of *Y*s, *X*s, or observations in the time dimension, the number of models that might be nested in this way becomes very large and the TSP program correspondingly larger. Using DOT loops and other shortcuts can make programming easier and streamline your program so that it is easier to read and debug. See the examples earlier in this chapter for ideas.

15.3.3 Dynamic factor models with panel data

An example of how to estimate a fairly complex dynamic factor model using SUR is available in the examples on the TSP web site. The example is drawn from Hall and Hayashi (1989).

15.3.4 GMM Estimation of panel data models.

A series of recent papers (Arellano and Bond 1991, Keane and Runkle 1992, Ahn and Schmidt 1992) have advocated the use of the GMM methodology for the estimation of dynamic panel models or panel data models with predetermined rather than exogenous right-hand side variables. These estimators are straightforward to implement in TSP using the GMM estimation command. In order to use this estimator on panel data, your data should be in panel format, and you should write a different equation for each year of data (this is easily done with DOT loops).

There are two ways to specify different instruments for each equation (year):

1. directly in the command using a special form of the INST statement where the instruments for to be used in each equation are separated by the "or" sign |.
2. using the MASK option to choose the instruments you wish to use for each equation.

As an example, consider the one-variable model of *Y* on *X*, with 3 years of data for each y, but 6 years, including 3 lags for each *X*. With this much data, it is possible to test not only for strong exogeneity of the *X*'s, but also for weak exogeneity of lag order 0, 1, or 2, either unconditionally or conditional on the presence of individual effects. That is, you can test for whether there is zero correlation between the first-differenced disturbances and future *X*'s and also for whether there is zero correlation between the first-differenced disturbances and current *X*'s, or *X*'s lagged once. As the number of lags of *X* variables that are not assumed to be exogenous increases, the number of moment restrictions imposed decreases. A simple TSP run that performs these tests conditional on individual effects is shown below:

```
? Equations of the model.
FRML UEQ86 Y86 - BETA1*X86-BETA2*X85-BETA3*X84 ;
FRML UEQ87 Y87 - BETA1*X87-BETA2*X86-BETA3*X85 ;
FRML UEQ88 Y88 - BETA1*X88-BETA2*X87-BETA3*X86 ;

? First-differenced versions.
FRML DUEQ87 UEQ87-UEQ86 ;
FRML DUEQ88 UEQ88-UEQ87 ;
DOT 87 88 ;
        EQSUB DUEQ. UEQ86-UEQ88 ;
ENDDOT ;
```

```
LIST XLIST X83-X88 ; ?  List of all instruments

?  Lag 1+ instruments in levels
GMM (INST=(X83 X84 X85 | X83 X84 X85 X86))
         DUEQ87 DUEQ88 ;
? Save chi-squared statistic for estimated model.
COPY @GMMOVID TRACE1 ;
? Save actual number of moment restrictions imposed.
COPY @NOVID DF1 ;

? Weak exogeneity – lag 0 instruments
GMM (INST=(X83 X84 X85 X86 | X83 X84 X85 X86 X87))
         DUEQ87 DUEQ88 ;
COPY @GMMOVID TRACE0 ;
COPY @NOVID DF0 ;
```

```
? Strong exogeneity – all instruments for all equations
GMM (HETERO,INST=XLIST) DUEQ87 DUEQ88 ;
COPY @GMMOVID TRACES ;
COPY @NOVID DFS;

? Compute Test Statistics.
SET TEST0 = TRACE0-TRACE1 ; SET DFR = DF0-DF1 ;
? Test lag 0 instruments, maintaining lag 1.
CDF(DF=DFR,CHISQ) TEST0 ;
SET TESTS = TRACES-TRACE0 ; SET DFR = DFS-DF0 ;
? Test strong exogeneity, maintaining weak.
CDF(DF=DFR,CHISQ) TESTS ;
```

Note that the first model estimated in this example is the least constrained model; all the others will be tested relative to this one.

15.3.5 Using the MASK= option

Instead of supplying separate instruments for each equation, you may prefer to use the MASK option to specify which instruments apply to which equation. For example, the first estimation above can also be done with these commands:

```
READ (NROW=6,NCOL=2) M1 ;
? Mask for lag 1 and greater instruments.
1 1
1 1
1 1
0 1
0 0
0 0
;
GMM (HETERO,INST=XLIST,MASK=M1) DUEQ87 DUEQ88 ;
```

Modifying this example to perform the tests without allowing for individual effects is straightforward: simply use the level equations UEQ86-UEQ8 and modify the M1 and M0 matrices accordingly. For example, if the order of the equations is UEQ86 UEQ87 UEQ88, the mask M1 should be the following matrix:

```
1     1     1
1     1     1
1     1     1
0     1     1
0     0     1
0     0     0
```

The examples section of the TSP web site http://*www.tspintl.com* contains more examples of estimating a panel data model using both the GMM and PI matrix techniques. There are also examples of performing LM tests for serial correlation as in Arellano and Bond (1991), and computing the one-step covariance matrix estimate recommended by Blundell and Bond (1995).

15.4 Panel data and nonlinear models

The implications of nonlinearity for panel data models depend on the way in which the individual effect enters the model. Two situations can be distinguished, one where the effect enters linearly and one that is intrinsically nonlinear:

$$E[y_{it} \mid X_{it}, \alpha_i] = f(X_{it}, \beta) + \alpha_i$$
$$E[y_{it} \mid X_{it}, \alpha_i] = g(X_{it}, \alpha_i, \beta)$$

The first case is easily handled by GMM via differencing to remove the effect and then instrumenting with lagged values of the independent (and possibly the dependent) variable, as described in the previous section. We discuss some alternatives for dealing with the second case (which includes count data and qualitiative depdendent variable models such as Probit) in the next two sections.

15.4.1 Multiplicative individual effects

A special case of the second model is the case where the effect is multiplicative:

$$E[y_{it} \mid X_{it}, \alpha_i] = \alpha_i f(X_{it}, \beta)$$

For example, this is the model usually implied by Poisson or negative binomial models for panel count data (see Blundell, Griffith, and Windmeijer 2002; Montalvo 1997). These authors show that models of this type imply the following orthogonality condition:

$$E\left[y_{it} - f(X_{it},\beta)\{ y_{i,t-1} / f(X_{i,t-1},\beta)\} \,|\, X_{i,t-1} \right] = 0$$

Because GMM handles nonlinear equations for the residual *u* in the same way as linear equations, it is straightforward to implemented such a model in TSP using GMM and the methods described in the previous section. Here we provide a simple example based on the data with 6 years for X and 3 years for Y that we used in that section.

```
? Equations of the model f(X,B).
FRML FXB86 BETA1*X86+BETA2*X85 ;
FRML FXB87 BETA1*X87+BETA2*X86 ;
FRML FXB88 BETA1*X88+BETA2*X87 ;

? First-differenced versions for multiplicative effects.
FRML DUEQ87 Y87-Y86*(FXB87/FXB86) ;
FRML DUEQ88 Y88-Y87*(FXB88/FXB87) ;
DOT 87 88 ;
        EQSUB DUEQ. FXB86-FXB88 ;
ENDDOT ;

? Estimate using instruments in levels corresponding to weak
? exogeneity (note that we drop one to reduce bias).
GMM (INST=(X83 X84 X85 X86 | X84 X85 X86 X87))
       DUEQ87 DUEQ88 ;

? Estimate using  Lag 1+ instruments in levels
GMM (INST=(X83 X84 X85 | X84 X85 X86))
       DUEQ87 DUEQ88 ;
```

15.4.2 Qualitative dependent variable models (Probit and logit)

The basic Probit model for panel data takes the following form:

$$Y^*_{it} = X_{it}\beta + \alpha_{\mathrm{i}} + \varepsilon_{\mathrm{it}}$$

$$Y_{it} = \begin{pmatrix} 1 \; if \;\; Y^*_{it} > 0 \\ 0 \; if \;\; Y^*_{it} \le 0 \end{pmatrix}$$

As in the case of the linear panel data model, the choice of estimator for this model depends crucially on whether or not you assume that the effects α_i are uncorrelated with the *X*'s. When the effects are not correlated, the model becomes a variance components version of Probit. In this case you have two choices: 1) estimate using the usual Probit estimator, which is consistent, and obtain grouped standard errors that are consistent in the presence of the implied variance-covariance structure of the disturbances; or 2) use PROBIT with the REI option to obtain maximum likelihood estimates under the assumption that the disturbance has the variance components structure implied by the model above.

Assuming that you have specified the sample via FREQ (PANEL) so TSP knows what the panel structure of your data is, the first approach is automatic when you estimate a Probit model using panel data. In this case, the option HCOMEGA=BLOCK is turned on by default, and grouped standard errors will automatically be computed.

To implement the "random effects" maximum likelihood estimation, include the REI option in your PROBIT command. The Probit random effects model estimated is the following:

$$y_{it} = \mathrm{I}(X_{it}\beta + \alpha_i + \varepsilon_{it} > 0)$$

$$\alpha_i \sim N(0,\sigma_\alpha^2) \quad \text{and} \quad \varepsilon_{it} \sim N(0,\sigma^2)$$

$$\sigma_\alpha^2 + \sigma^2 = 1$$

$$\rho = \frac{\sigma^2}{\sigma_\alpha^2 + \sigma^2}$$

The normalization above means that the coefficients are normalized the same way as the results from the usual PROBIT command. The parameter ρ (RHO) is estimated and corresponds to the share of the variance that is within individual. The likelihood function involves computing a multivariate integral and this is done with Hermite quadrature, using a default 20 points; when RHO (the within variance share) is high, it may be necessary to increase this using the NHERMITE option.

Fixed effect Probit models are more problematic, since the usual dummy variable is inconsistent (suffers from finite T bias). For a full discussion of this issue, see Chamberlain (1982, 1984) or Wooldridge (pp. 482-492). Nevertheless, such models can be useful if T is even moderately large, since the bias is typically of order *1/T*. For this reason, TSP does provide a fixed effects Probit estimator via the FEI option. To use this option, FREQ (PANEL) must be in effect. A very efficient algorithm is used, so large unbalanced panels can easily be handled.

If desired, you can use the FEPRINT option to print a table of the effects, their standard errors, and t-statistics. Individuals that have dependent variable values that are all zero or all one are allowed, although their data are not informative for the slopes. The fixed effects for such individuals will be either a very large negative number (in the case of a zero dependent variable) or a very large positive number (in the case of one). These values yield the correct probability for these observations (zero or one).

As Chamberlain (1984) pointed out, and as is discussed in Wooldridge, the fixed effects logit model does have an estimator that is consistent for the slope parameters. This estimator can be obtained using the conditional density approach that was used in Hausman, Hall, and Griliches (1984) to obtain a consistent estimator for the Poisson model with individual correlated effects. Using the earlier notation but allowing ε_{it} to have a Type I extreme value distribution rather than a normal distribution, the likelihood for this model conditional on the sequence of 0-1 binary variables actually observed for individual *i* is similar to but not the same as the usual conditional logit likelihood:

$$\log L_i = \sum_{t=1}^{T} y_{it} X_{it}\beta - \log\left(\sum_{d_t \in B_i} \exp\left[\sum_{t=1}^{T} d_t X_{it}\beta \right] \right)$$

$$\text{with } B_i = \left\{ (d_1,...,d_T) \text{ such that } d_t = 0 \text{ or } 1 \text{ and } \sum_{t=1}^{T} d_t = \sum_{t=1}^{T} y_{it} \right\}$$

That is, each observation is conditioned on all the sequences of 0-1 binary variables that sum to the observed sum for the individual. Individuals whose observations are all 0 or all 1 do not contribute to this likelihood, since they convey no information about the relationship between X and the outcomes

Here is a simple example for the case when T=3. In this case the possible sequences of the dependent variable are (0,0,0), (0,0,1), (0,1,0), (0,1,1), (1,0,0), (1,0,1), (1,1,0), and (1,1,1). Any observations with the first or last sequence are dropped, leaving 6 sequences possible, 3 whose sum is 1 and 3 whose sum is 2. Assuming that the model contains only a single X and that all the data for an individual is in a single observation (*panel* format), the following statements set up the likelihood function for estimation:

```
sumd = d1+d2+d3 ;   ? Create the sum of the 3 dep vars
?    Dummies corresponding to the four possible sequence sums
dummy sumd a0 a1 a2 a3 ;
?   Choose the cases where sumd is 1 or 2 for estimation.
select a1|a2 ;

dot 1-3 ;
  frml xb. x.*beta ;
enddot ;

?   Note the use of a1 and a2 to select the appropriate denominator
?   for the likelihood.
frml logit d1*xb1 + d2*xb2 + d3*xb3
     - a1*log(denom1) - a2*log(denom2) ;
frml denom1 exp(xb3) + exp(xb2) + exp(xb1) ;
frml denom2 exp(xb2+xb3) + exp(xb1+xb3) + exp(xb1+xb2) ;

dot 1 2 ;
  eqsub denom. xb1-xb3 ;
enddot ;
eqsub logit denom1 denom2 xb1-xb3 ;

param beta 3 ;
ml (hiter=n,hcov=nbw) logit ;
```

16. STORING DATA ON EXTERNAL FILES

The free-format data loading described in Chapter 3 is easy and convenient to use, but it may not be suitable for large amounts of data: the data may not be in a form that can be read this way; it is somewhat inefficient to use free format; or you may have a large number of series in your data set, only a few of which you wish to use at one time. TSP provides several alternative formats for the storage and use of data in external files.

These alternate file formats are generally used in different situations and for different data problems. In this chapter, we try to give some guidelines for their use. Here are some possible situations:

1. You have a moderate number of not very long series (approximately 50 observations or fewer). You could type the data into a free format file, or a spreadsheet file (**.wks**, **.xls**, etc., but not spreadsheet notebooks).

2. Your project involves a large amount of data: either panel data with several hundred observations on a few time periods for each variable, or a large cross section (more than one thousand observations). Cost considerations dictate that data storage and input/output be as efficient as possible. In this case, the most efficient technique is to store the data in a TSP databank, after reading it into your TSP program. Another efficient alternative is to store the dataset in **stata** (**.dta**) format. You can do this either with **stata**™ or by using **stat/transfer** to convert from another format to stata format.[26]

3. You are building a large model section by section: the total number of series involved is quite large, but they may have different starting dates and frequencies. This kind of project is also well suited to a databank. You can easily store the documentation for each series along with the data values.

The next few sections discuss data storage alternatives in more detail and give a few examples of using them.

[26] See http://www.stata.com and http://www.stattransfer.com for information about these programs and how to obtain them. Note that variable documentation in stata format files will be saved as documentation when they are read into TSP, and will therefore be stored in any TLB files that you create.

16.1 Using external sequential files: READ, WRITE

A *sequential* file is just what its name implies: one record follows the next, in sequence. The file is always written and read in a fixed order. In TSP, the reading and writing of data to an external sequential file are done with the READ and WRITE commands, using the FORMAT= and FILE= options. The FORMAT= option tells TSP whether the data on the external file is free format, formatted (characters), binary, or a special format such as spreadsheet.

In a *formatted* data file (sometimes called an ASCII file after the character code used or a text (.txt) file), each number occupies a prespecified number of spaces, and is represented by a series of characters in a particular machine code representation. The advantage of formatted data is that it can be printed easily and is readable on different kinds of computers.

Binary or *unformatted* data, on the other hand, is not readable on a different kind of computer than the one on which it was written, nor can it be printed directly. It is the most efficient and the least flexible form of storage The format of this data is the same as the machine representation of the numbers; so no recoding is done as it is read or written and maximum efficiency is achieved.

For further information on the use of formatted and unformatted data see FORMAT in the *Reference Manual*.

The FILE= option is used to specify the external file from which the data will be read or to which the data will be written.

```
READ(FILE='FOO.DAT') X Y Z;
```

specifies that TSP is to read the series X, Y, and Z in free format from the file FOO.DAT.

16.1.1 Data organization on external files

You can organize data on an external file by series or by observation. Data organized by series typically has one or more records per series, with all the observations, and starts a new record for each series. Data organized by observation is more typical of cross-section research: each record contains all the variables for a single observation and each observation starts with a new record. TSP can read/write data stored in either way. For example, to load data stored by series:

```
FREQ A ; SMPL 48 79 ;
READ(FILE='macro.dat',BYVAR) GNP IMPT RELP ;
```

The external file macro.dat has 32 numbers for GNP (possibly on more than one line), then 32 numbers for IMPT, and 32 numbers for RELP.

The next example shows how to read data stored by observation: each pair of records on the input file consists of data on the 11 variables being read; there are 385 pairs of records corresponding to the SMPL of 385 observations:

```
FREQ N ; SMPL 1 385 ;
READ(FILE='PATENTS.DAT',FORMAT='(7F10.5/10X,4F10.5)') ID
        PATENTS LRNDL5 LRNDL4 LRNDL3 LRNDL2 LRNDL1 LRNDL0
        TIME BOOK71 SCISECT ;
```

Suppose that you now wish to reformat the data in the second example as a TSP databank file for efficiency in storage and reading. You can use OUT and KEEP statements to do this:

```
OUT PATDATA;
KEEP ID PATENTS LRNDL0 LRNDL1 LRNDL2 LRNDL3 LRNDL4
        LRNDL5 TIME BOOK71 SCISECT PFIT U ;
```

In this example we assume that you also wished to add two computed variables (PFIT and U) to the databank PATDATA. On most computers, the new file has approximately 20,000 bytes, whereas the old formatted file had about 54,000 bytes, even though it had fewer variables. An easier way to save all original and transformed variables is to put the OUT command before the READ command; then the KEEP command is not needed.

In dealing with large cross-section data files organized by observation, you often do not know the exact number of observations in the file before you make the first run. The READ command has a convenient option to solve this problem, called SETSMPL. SETSMPL does exactly what its name implies: it sets the SMPL after the data is read and the number of observations is known. This option is the default if the SMPL has not yet been defined. For example if PATENTS.DAT was actually a free format file:

```
READ(FILE='PATENTS.DAT') ID PATENTS LRNDL0 LRNDL1 LRNDL2
        LRNDL3 LRNDL4 LRNDL5 TIME BOOK71 SCISECT ;
```

After these commands have been executed the SMPL will be 1 385.

In cases where you wish to use a frequency other than N, provide a FREQ statement and a SMPL with the correct starting date of your file, and then use the SETSMPL option explicitly in the READ statement to update the ending date of the SMPL from the data you input.

16.1.2 Closing external files: CLOSE

Files are opened when they are first referenced in a READ or WRITE statement and are not closed until the end of the program, unless one of the following two things happen:

1. The same filename (or unit) is used in both READ and WRITE statements.
2. Many different files (usually more than 12) are used in a single program.

In these situations, the CLOSE command is available to give the user explicit control over the closing of files. CLOSE uses the option FILE=, just like READ and WRITE. Most users will never have to use the CLOSE command, since the default methods are adequate for most file operations.

16.2 Spreadsheet files

TSP can read series and matrices directly to or from spreadsheet files. The following table lists the file types supported by TSP:

Spreadsheet	**Version**	**Filename extensions**	**TSP support**
Lotus 123, Symphony	1,2	.wks .wk1 .wrk .wr1	Read, Write
Lotus 123	3	.wk3	Read
Lotus 123/J (Japanese)	1,2	.wj1 .wj2 .wk2 .wt2	Read, Write
Microsoft Excel	2	.xls	Read, Write
Microsoft Excel	3,4	.xls	Read
Microsoft Excel	5,97,2000, 2002	.xls	Read/Write
Quattro Pro		.wq1	Read (PC only)

TSP generally writes the oldest file formats, which are always readable by more recent spreadsheet releases. The exception is Excel where TSP usually writes in the format of version 4.0.

Spreadsheet files should take the format of the following example, which consists of quarterly data from 48:1 to 49:1:

	A	B	C
1	Date	CJMTL	PMTL
2	Mar-48	183.4	#N/A
3	Jun-48	185.2	.436
4	Sep-48	192.1	.562
5	Dec-48	193.3	.507
6	Mar-49	206.9	.603

Series are read from individual columns in the file. Series names are optionally supplied in the first row of the file above the data columns. Dates may be given in the first column. Many different file configurations are possible, and it is possible to read in some series while ignoring others. Following are some simple guidelines for creating a spreadsheet file for TSP:

1. Put the column names in the first row. They should be valid series names in TSP (lowercase is fine, but imbedded blanks and special characters are not allowed).[27] If the file has no names, you can supply them when you read the file into TSP, but this can be inconvenient. If the file contains invalid names, the data can be read by TSP as a matrix, ignoring the current names, and you can supply your own names inside of TSP. TSP will not recognize names in lower rows or in sheets after the first. TSP treats these names as missing values and numbers below them will be read as data for the original column names.

2. The second row must contain data.

[27] Unlike UNIX, TSP makes no distinction between lower and upper case when reading commands and variables, but all variable and file names are stored in upper case, which means that on UNIX and in unix-style programs like stata you will need to be aware that the names are uppercase.

3. If you are reading time series, the first column should contain dates. This will ensure the series are read with the proper frequency and starting date, regardless of the current FREQ and SMPL in TSP. Dates can be strings such as 48:1 or numbers formatted as dates (Mar-48, 3/31/48, etc.). You only have to supply enough dates so that TSP can detect the frequency (5 is enough to distinguish between quarterly and monthly). TSP ignores any dates after these, assuming that the data is contiguous (no missing periods/years, or SMPL gaps in TSP terminology). If you have missing periods/years for all series, leave the corresponding rows blank. Below is a table of examples showing recommended ways of defining the starting date and frequency with a dates column. If you have dates in other columns, they will be read as numbers. If you are reading a matrix, the date column will be ignored (i.e. it will not be read into the matrix).

First date	**Second date**	**Resulting frequency**
string dates – first character is ‛ ^ or ‟		
1	2	A if current freq is A, otherwise N
48:	Any	A
48:5	Any	M
48:4	49:1	Q
1948:1	1948:2	M or Q depending on subsequent dates
48:1	49:1	A
A2:3	Any	Invalid
48:2	48:1	Invalid
numeric dates		
12/31/48	12/31/49	A (any dates 365-366 days apart)
12/31/48	1/31/49	M (any dates 28-31 days apart)
12/31/48	3/31/49	Q (any dates 90-92 days apart)
12/31/48	1. 1.49	N (any other date range)

4. Missing values can be represented by blank cells or by formulas which evaluate to missing.

To read in a spreadsheet file, use the FILE='filename' option. FORMAT=LOTUS or EXCEL is optional. If the filename contains one of the extensions listed earlier (.WKS, .WK3, .XLS, etc.), TSP checks the first few bytes of the file to confirm that it is one of the spreadsheet versions listed above. Conversely, if FORMAT=LOTUS or EXCEL is specified, but the filename does not contain a extension , then .WKS or .XLS is appended to the filename. To read a matrix (bypassing column names and dates), use the TYPE=GEN option. TYPE=CONSTANT is not supported for Lotus files; series will be defined instead. The NCOL=, NROW=, IFULL, UNIT=, etc. options are ignored, but SETSMPL is supported.

If no series names are supplied on the READ command, TSP looks for column names in the file and creates series with those names. If you supply series names, TSP attempts to match them to column names in the file. If the file does not have column names, you must supply a READ argument for each data column. If you are unsure of the file's contents, check it with your spreadsheet editor or read it as a matrix. If you think the file has column names, but you don't know what they are, try supplying a dummy name that won't be matched and TSP will print an error message listing the column names in the file. If you are reading a matrix (using TYPE=GEN as mentioned above), TSP will create a matrix named @LOTMAT unless you supply an argument (the name of a matrix) to READ.

If for some reason your series are in rows instead of columns, you can read the file as a matrix, transpose it, and UNMAKE the matrix into series. Alternatively you could transpose the file in your spreadsheet editor: most have a special paste option for performing this operation (e.g., Excel).

TSP checks the first row in the file for string names (Lotus cells beginning with the characters ' ^ or "). The names are truncated to 8 characters (if necessary), and are translated to uppercase. They must be aligned above their corresponding data columns. If you have dates in the first column, no name is required for the date column. Any names with imbedded blanks will be ignored.

READ examples for spreadsheet files

1. We will use the SML.WKS file shown below in the following examples:

	^CJMTL	^PMTL
'48:1	183.4	NA
'48:2	185.2	.436
'48:3	192.1	.562

'48:4	193.3	.507
'49:1	206.9	.603

```
READ(FILE='SML.WKS');
? the series CJMTL and PMTL are defined. FREQ Q and
? SMPL 48:1, 49:1 are set if there is no current FREQ or SMPL.

READ(FILE='SML.WKS',TYPE=GEN);
? creates the 5x2 matrix @LOTMAT, with the values
? of CJMTL and PMTL in its columns.

READ(FILE='SML.WKS') PMTL;          ? only reads in PMTL
```

2. Here is the nm3.wk1 file (as shown in Lotus, using numeric dates) for the following examples:

04/30/57	23.2	34.5	10.9
05/31/57	23.6	35.1	11.0
06/31/57	23.9	35.8	11.2
07/31/57	24.0		11.5

```
READ(FILE='NM3.WK1') SF LA SD;
```

defines the monthly series SF, LA, and SD from 57:4 to 57:7. If there is no current SMPL, this is the new SMPL with FREQ M. The series LA will be given a missing value in its last observation.

```
READ(FILE='NM3.WK1') SF;
```

causes an error message to be printed because three series names are required unless the option TYPE=GEN is supplied (to indicate that the data should be read as a matrix).

16.2.1 Writing spreadsheet files

The WRITE command will write either a single matrix or several series to a spreadsheet file. If the file specified already exists, it is overwritten by the new file. If you write a matrix, the file will consist only of numbers. If you write series, their names will be put in the first row. The first column will contain dates (or observation numbers), and each series will be put in a column below its name. Series are written under the control of the current sample. If there are gaps in the sample, observation numbers will be used instead of dates in the first column. Dates are written as the last day of each period, and formatted as Mon-Yr.

WRITE examples for spreadsheet files

```
? This creates a file equivalent to the READ examples, except that
? dates are formatted numerically instead of as strings.
FREQ Q;
SMPL 48:1,49:1;
READ CJMTL; 183.4 185.2 192.1 193.3 206.9;
READ PMTL; . .436 .562 .507 .603;
WRITE(FILE='SML.WKS') CJMTL,PMTL;

? This creates a file with the same data columns, but no dates or
?  series names (since a matrix is used).
FREQ Q;
SMPL 48:1,49:1;
LOAD CJMTL; 183.4 185.2 192.1 193.3 206.9;
LOAD PMTL; . .436 .562 .507 .603;
MMAKE M CJMTL,PMTL;
WRITE(FILE='SMM.WKS') M;
```

Here is the resulting SMM.WKS file:

	A	B
1	183.4	NA
2	185.2	.436
3	192.1	.562
4	193.3	.507
5	206.9	.603

16.3 TSP Databanks

TSP databanks provide a convenient means of automatically storing and retrieving your variables from disk.

The method for storing and retrieving variables from databanks is implicit. Instead of listing variables in a READ statement, the names of databanks are listed in an IN statement. Any variables in the databanks are then available, and are read in automatically if referenced. The corresponding statement for storing variables is OUT; which marks all variables created or modified after it for output to files listed on the statement. For example,

```
FREQ A; SMPL 48,87;
IN USNIPA; OUT USNIPA;
OLSQ GNP C,CONS;    ? old series read from USNIPA databank
LGNP = LOG(GNP); LCONS = LOG(CONS);
? new series saved in USNIPA
```

reads in GNP and CONS from databank USNIPA, runs a regression using them, takes their logs, and add the new logged variables to the same databank.

16.3.1 Storing variables in a databank: OUT

Variables created or modified during a TSP run will be stored as members of a databank at the conclusion of the run if you include a statement of the following form in your TSP run:

```
OUT FILENAME ;
```

After an OUT statement, all TSP variables that were created or modified are written to the file associated with that filename. The variables stored can include series, constants, matrices, or equations, provided they have been created or changed after the appearance of the OUT statement. The example above would create a file called FILENAME.TLB.

A new OUT statement later in the run will cause all items created after its appearance to be put on the file(s) named on it, but will leave the items stored previously on the first file(s) unchanged. The statement

```
OUT ;
```

causes TSP to cease storing anything on the output files. Here is an example:

```
OUT DB1 ;
PRINT W1 LNX1 ; GENR GNP=G/P ;
OUT DB2 ;
GENR W1ACT=W1 ; GENR W1=A1+B11*LNX1 ;
OUT;
GENR W2=A2+B22*LNX2 ;
END ;
```

After this run is completed, the series GNP will be stored in DB1.TLB, W1ACT and W1 will be stored in DB2.TLB, and W2 will be stored on neither file.

Occasionally it is convenient to store variables during a TSP run without having to create or modify them. To do this, use a KEEP statement:

```
KEEP VARNAME1,VARNAME2,.....;
```

This puts the variables named VARNAME1, VARNAME2, etc. in the file referenced by the current OUT statement. This is the only way (other than WRITE(FORMAT=DATABANK,...)...;) to store equations and @ variables in a databank. Standard OUT commands ignore these variables. Only LISTs and PROCs cannot be stored in databanks at all. The KEEP command below stores the equation EQ1 and the variance-covariance matrix @VCOV in the databank FILE1.TLB in addition to the series X and Y and the estimated parameters A and B (which are automatically stored).

```
FREQ A ; SMPL 66 75 ;
OUT FILE1 ;
LOAD X ; 11 9 14 30 18 12 29 35 31 25 ;
LOAD Y ; 1 2 3 4 5 6 7 8 9 10 ;
FRML E1 Y=A+B*X ;
PARAM A B ;
LSQ E1 ;
KEEP E1 @VCOV ;
END ;
```

If you want to save all the series and variables in your TSP program, use the command KEEP ALL.

16.3.2 Documenting variables on a databank: DOC

Since TSP databanks are permanent storage for your data, and may be kept for several years and updated, it is useful to store documentation about your data with the actual series. Otherwise, it is easy to forget exactly what they are if you leave a project for a few months and then return to it. TSP provides the DOC command to let you store information about variables in your databanks. For example, to store descriptions for the preceding example, include these statements somewhere after the OUT FILE; statement:

```
DOC X 'The exogenous variable from the DBRUN example' ;
DOC Y 'The endogenous variable from the DBRUN example' ;
DOC E1 'The equation from the DBRUN example' ;
```

There is no restriction on the length of the documentation that you can store. Also, any variable documentation in stata files that you input to TSP will automatically be stored.

16.3.3 Retrieving variables from a databank: IN

When a TSP databank has been created, it can be accessed by including a statement of the form

```
IN FILE1,FILE2,...;
```

If there is more than one filename in an IN statement, each file will be searched until the relevant variable is found. Up to 8 filenames may appear on one IN statement. When another IN statement is encountered with new filename(s), TSP will stop searching the first group of files and search the new group for any new items encountered after the second IN statement. The statement

```
IN ;
```

will cause the program to cease searching any files until a new IN statement is encountered.

The following simple job shows the minimum requirements for accessing the databank created by TSP in the last example in the previous section:

```
NAME DBRUN 'IN DATABANK STATEMENT' ;
IN FILE1 ;
```

```
PRINT X Y ;
```

Note that it is not necessary to set the FREQ and SMPL before the first reference to the databank; normally TSP will obtain these from the first series in the databank if they are not already specified. Since each series is stored with its own frequency and starting date, series of different frequencies may be mixed in a databank, but this is not recommended.

16.3.4 Databank utilities

If you forget what a databank contains, you can use TSP commands to show a databank's contents, copy a databank, and delete variables from it.

DBLIST prints the variables in a databank, with their associated lengths. The information is printed in the SHOW command format, which lists starting/ending dates for series, matrix dimensions, and values for scalars.

DBPRINT prints the values for all of the series in a databank, using the current SMPL and FREQ. If you are not sure which SMPL and FREQ to use, check the databank first with the DBLIST command. Of course, standard IN and PRINT commands can be used to list selected variables from a databank.

DBCOPY creates a TSP program file from a databank. This file contains all the commands and data necessary to create the databank on another computer. This can be used to move databanks between different types of computers (such as from PC to mainframe), or as an alternative to DBPRINT for listing the values of non-series variables.

TSP also provides the command DBDEL which can delete one or more variables from a TSP databank. DBDEL can erase variables which were put in the databank by mistake, or which are no longer needed. It can also be used to rename variables in a databank, by first copying the variables to new names and then deleting them using the old names.

16.3.5 Using older databanks in TSP 5.0

Since version 4.2, TSP has contained the capability to store documentation with variables, and store scalars in double precision. This means that databanks older than 4.2 must be converted in order to be used. The conversion is automatic; the old databank will be renamed from BANK.x to BANK.T41 (or BANK.T35 if it was from TSP Version 3.5), and the new databank will be called BANK.TLB. If you are converting Version 3.5 databanks, you have to specify the appropriate FREQ and SMPL before conversion.

16.3.6 Using Eviews/micro-TSP databanks in TSP

Eviews^{TM} (formerly **micro-TSP)** is the econometrics package written and sold by Quantitative Micro Software; it is similar to, but not the same as TSP. **Eviews/micro-TSP** databanks have a different format from TSP databanks, but TSP can read them.

Eviews stores only one series in each of its databanks, using the series name with the extension .DB. The databank files are not written in binary, and are not very efficient, but they can be edited easily with a text editor and can include comments. TSP has FETCH and STORE commands to access this type of databank (the standard TSP IN and OUT commands will not work with these databanks). These commands should be useful if data is already available in this format, or if this format is preferred over the more efficient standard TSP databank.

For example, to read two series from **Eviews** databank files X.DB and Y.DB and run a regression:

```
FETCH X,Y; OLSQ Y C,X;
```

The equivalent operation, using a standard TSP databank file US.TLB is:

```
IN US; OLSQ Y C,X;
```

16.4 Saving a work session in a file: SAVE, RESTORE

When using TSP interactively (see Chapters 4 and 17), it is often useful to save all current variables in a file, exit from TSP, and then resume the session later. To do this use the SAVE and RESTORE commands. SAVE creates the binary file TSPSAV.SAV, which contains values for all the current variables (series, matrices, scalars, FRMLs, etc.). RESTORE reads this file and essentially re-creates the session as it was when the user exited from TSP. (The command stream, including PROCs, could also be restored by using INPUT with a renamed BKUP.TSP file).[28]

An name other than TSPSAV can be specified as an argument to SAVE or RESTORE, but the resulting file will always have the .SAV extension. SAVE and RESTORE are not intended for permanent storage of data. Databanks are preferred, since they are more flexible. Databanks allow access to individual variables without having to bring all the variables into memory, and can be incrementally updated.

[28] Note that there is no compatibility between **TSP** and **Eviews/micro-TSP** SAVE and RESTORE files.

17. TIME SAVERS IN INTERACTIVE TSP

This chapter outlines interactive TSP features that save you time and make the program more convenient to use. Some of these, such as re-executing modified commands, you may use every interactive session; others, like automatic backup and recovery, are "there when you need it". See the *Reference Manual* for further information on all the commands described in this chapter.

Below the interactive and editing commands available are described in some detail. However, you should be aware that there are two main ways to edit and re-execute commands in the interactive version of TSP: one, which is used in PC and Macintosh versions, uses the familiar cursor keys (arrows, backspace, delete, page up, page down, home, and end) to find and edit commands. The second, which is used in mainframe and unix versions, relies on the older and more awkward line-oriented methods. We describe both below, but most users will find the cursor key methods easier to use when they are available (which is most of the time). In the next sections, we refer to these as the “PC method.”

17.1 Revision and re-execution of commands

Interacting with a program implies that your input commands may depend on the results previously output. Often this only involves modifying your previous command. You might want to fix a typo, change a list of options, add a term to an equation, etc.... In any case, you are spared retyping the whole command. Interactive TSP offers several ways to do this. Note that the following sections apply to *commands* only; data should be modified with UPDATE.

17.1.1 Re-execution

There will be occasions when you want to execute a command or group of commands entered previously without performing a modification. There are two ways to do this. One (the PC method) is to simply hit the “page up” key until you see the command you want to re-execute and then type return.

The second, available on mainframes and workstations running unix uses the EXEC command. For example, typing

```
EXEC 10
```

causes the re-execution of line 10 in your program. If two line number arguments are given, the range of lines between them (including them) will be executed.

Some examples of when EXEC might be used:

1. Execution was initially suppressed with an EXIT command in while in collect mode or in an input file.
2. Errors were discovered in the data and results need to be regenerated after an UPDATE has been performed.
3. Commands you use more than once and don't want to retype (for example, a SMPL with lots of gaps).
4. A modification to the original command was performed separately using EDIT (see below).

The use of EXEC in collect mode is slightly different, and is discussed in the *Reference Manual*.

17.1.2 Modifying commands and fixing typos

In Chapter 4, we introduced the easiest way to revise commands and re-execute them when you are interacting with TSP: editing using the cursor keys (the PC method). However, there are times when you would like to simply add or delete a variable name, or re-execute a whole set of commands as a group. For these purposes, you may find the interactive commands EDIT, RETRY, ADD, and DROP useful. In this section, we describe EDIT and RETRY. In the next section, we describe ADD and DROP, which are special cases of RETRY.

Use EDIT and RETRY when you want to change the arguments in a command. EDIT makes the changes without executing them. RETRY combines EDIT, which makes changes, and EXEC, which executes them. Syntax, "arguments" and editing rules are explained in the *Reference Manual*, and are identical for both EDIT and RETRY. Here are some examples of using EDIT:

```
EDIT 10
```

would request modification of line 10; the line will be displayed, and the edit prompt issued:

```
10. CONS C GNP
>>
```

If no line is requested, the previous line is assumed. Only one line may be edited at a time (i.e. only one line number argument will be accepted). The basic editing functions are insert, delete, and replace.

Possible uses for EDIT are:

1. Modifying a statement that is a definition rather than executable (for example, an equation definition using FRML).
2. Modifying a range of lines before re-executing them as a group, or modifying a line within a range that will then be executed as a group.
3. Correcting mistakes made while entering commands in collect mode.

RETRY is appropriate when:

1. Correcting typos in interactive mode.
2. Re-executing a command with a modified list of options or variables.

The following example illustrates the relationship between EDIT and RETRY, as well as basic use of the editor:

```
10? INT DP,DP1,LGNP,TIME,C INVR C,G,LM,TIME

<error message because procedure INST is misspelled>

11? EDIT 10
>> REPLACE INT INST 1
>> EXIT

10. INST DP,DP1,LGNP,TIME,C INVR C,G,LM,TIME ;

12? EXEC 10
10. INST DP,DP1,LGNP,TIME,C INVR C,G,LM,TIME ;

<output from the INST command>
```

This same change can be performed more easily with the RETRY command (note that we have used all possible abbreviations and omissions to save typing).

```
11? RET
>> R INT INST
>>
10. INST DP,DP1,LGNP,TIME,C INVR C,G,LM,TIME ;
```

<output from the INST command>

17.1.3 Adding and dropping variables

ADD and DROP are used to add or drop variables from the previous command and automatically re-execute it. They simplify the task of re-executing the previous estimation command with a modified list of series, but are not limited to that particular use.

ADD and DROP both make permanent modifications to the "previous" command. The "previous" command, is defined as the last TSP command that *is not itself* ADD or DROP.

```
ADD var1 var2
```

is identical to

```
RETRY
>> INSERT var1
>> INSERT var2
>> EXIT
```

Both insert var1 and var2 at the end of the previous command (presuming there was not a sequence of ADDs and DROPs) and execute it. Similarly,

```
DROP var1 var2
```

is identical to

```
RETRY
>> DELETE var1
>> DELETE var2
>> EXIT
```

Both delete the first occurrences of var1 and var2 in the previous command (same assumptions) and execute it.

It is not possible to combine ADD and DROP into one step to perform a REPLACE function, or to make compound modifications. In these circumstances, the RETRY procedure or up-arrow editing must be used.

17.2 Reading commands from disk

An intermediate approach to running TSP programs which combines features of batch and interactive execution is provided by the INPUT command. This procedure reads a TSP command file from disk and executes it in TSP; following its execution you are left in interactive mode with all the data and results available for use.

The command

```
INPUT filename
```

will read filename.tsp in the current directory and execute it. All commands read are added to the session log, and may subsequently be REVIEWed, EDITed, etc....

You will also be given a choice as to whether you want the output directed to the terminal or to an output file. Creating an output file this way differs from the OUTPUT command in two ways: 1) if the output file already exists, a new version will be created (possibly overwriting the old), and 2) the file will automatically be closed at the end of execution of the input file (you may still append to it with the OUTPUT command).

An input file may be an entire TSP program prepared for batch execution or groups of TSP commands frequently executed as a unit. Advanced users might find it convenient to keep a library (i.e., directory) of TSP PROCs for use as input files. Since a PROC is not executed until it is "called", these files would produce no output when read -- adding a "SHOW procname" command at the end of each would provide quick information on the PROC just defined. Input files may be nested up to a depth of 5.

Reading an input file is exactly the same as using collect mode -- the difference is the source of the command stream. All commands will be read until execution is "requested"; for an input file this means encountering an END statement, or end-of-file. Just as in collect mode, execution may be suppressed by replacing the END statement with EXIT; you may then REVIEW what was read, and EXEC just the sections you need.

17.2.1 Your TSP login file

Every time you begin an interactive (or batch) session, TSP looks for a file called LOGIN.TSP before prompting you to begin entering commands. This file is a special case of an input file. The only difference is that execution is automatic at startup, and you will not be given the option of directing any resulting output to a disk file. If LOGIN.TSP does not exist in the current directory, your home or root directory will be searched as well.

If much of your work with TSP uses the same data, you may want to use LOGIN.TSP to set the SMPL and FREQ, load data, or open databanks. It is also convenient to place the OPTIONS command here.

17.3 Talking to the operating system (DOS or unix)

When you are using the interactive version of TSP you may wish to interact with the operating system. On multitasking windowing systems like Windows and Mac OS, this is done in the usual way. For DOS or unix users, TSP offers the SYSTEM command, which allows you to suspend your session temporarily, take care of other business on the system, and pick up later right where you left off. SYSTEM takes no arguments, and simply produces the message

```
Enter commands. Type CONTINUE to resume TSP session.
$
```

you may now create or modify files, or enter other commands. You may enter commands as long as you like;

```
$ CONTINUE
```

will resume your interactive session with no loss of continuity. There are many uses for this feature, one of the most useful is the ability to fix data files or examine output files created during your session without halting the program. Please note that in order to use it in this way, the files in question must be closed first with the CLOSE, TERM, or OUTPUT command.

17.4 Automatic backup and recovery

Interactive TSP saves a copy of the input of your session when you exit the program, and if your session is terminated abnormally this copy provides the ability to recover the session.

During an interactive session, TSP automatically saves the command stream in a file named INDX.TMP. INDX.TMP is used by commands like EDIT which modify the command stream. INDX.TMP is written in a special format for this purpose. Upon normal exit from TSP, INDX.TMP is used to create the standard sequential file, BKUP.TSP, found in your directory and INDX.TMP is deleted.

BKUP.TSP is useful for reconstructing a session and can be run by TSP in batch mode, since interactive commands have been stripped from it. Note, however, that data entered with the ENTER and UPDATE is not saved in BKUP.TSP -- data should be saved in a databank or a SAVE file (see Chapter 13).

If you accidentally (or purposely) type CTRL-C, TSP will terminate abnormally. The same is true if the program encounters a fatal Fortran error or system failure. In these cases BKUP.TSP will not be created. However, if you reenter the program, TSP will be able to recover your session from the file INDX.TMP which still remains on disk. Only commands will be recovered -- you must EXEC portions whose results you want reinstated.

If TSP finds a file INDX.TMP on disk when you start the program, you will get the message:

```
WARNING> Your previous TSP session was terminated abnormally.
Do you wish to recover it (y/n)? [y]
```

If you have changed directories or accounts since termination, or renamed INDX.TMP, TSP will not know you need to recover the session. You must then request recovery with the RECOVER command. Rules for specifying a filename are the same as those for the INPUT and OUTPUT commands.

Important note: Do *not* try to use INDX.TMP as an INPUT file, or edit it (you will destroy the special format). Conversely, do not try to RECOVER any files that were not originally an INDX.TMP file created by TSP, because other files will not have the correct format for recovery.

A. BASIC RULES OF TSP

This appendix summarizes the basic rules of the TSP language. A table at the end lists the complete TSP character set and each character's interpretation or legal use.

A.1 Rules for composing TSP names:

Every name must begin with a letter, _ # % or @ (exceptions: 2SLS and 3SLS commands).

Subsequent characters in a name may be letters, _ # % @, or digits.

# or % may not be used in names that appear in MATRIX commands.

The maximum number of characters permitted in a name is 64 (in versions prior to TSP 4.4 it was 8).

A.2 Rules for composing text strings:

A text string must be enclosed by matching pair of quotes (" or ').

Quotes are allowed in a string when they are of a different type from the enclosing quotes, i.e. "Can't" or '"sometimes"' (interpreted as Can’t and “sometimes”).

A.3 Rules for composing numbers:

Every number must begin with a ., +, -, or a digit.

No spaces may appear within a number.

One decimal point may appear.

One E or D may appear followed immediately by a one or two digit number with or without a sign. This is interpreted as a power of 10 to multiply the first number.

Example:

```
1E2 = 100
```

With free-format LOAD or READ commands, a . is interpreted as a missing value, and a repeat count with a * may be specified.

Examples:

```
3*0 is treated as 0 0 0
53 . 100 is treated as 53, missing, 100
```

The largest value of a series (in absolute value) that may be stored in TSP is 1.E-37, unless OPTIONS DOUBLE ; is used. Values larger than this are set to missing. Scalars and matrices are always stored in double precision.

A.4 Rules for composing algebraic formulas:

In general, TSP rules for formulas are similar to Fortran or other scientific programming languages.

A lag is indicated by putting an integer or a name in parentheses after a series name. The integer is negative for lags and positive for leads. A + sign is not necessary for leads. If the lag or lead is a name, it must have no more than four characters.

A series may have a single numeric or variable subscript (or lag/lead). A matrix may have a single or double subscript (numeric or variable). See the SET command for detailed rules and examples.

Arithmetic operators are

+	add
-	subtract
*	multiply
/	divide
**	raise to the power

Functions are the following:[29]

LOG()	Natural logarithm
EXP()	Exponential function
ABS()	Absolute value
LOG10()	Log base 10
SQRT()	Square root
SIN()	Sine (argument in radians)
COS()	Cosine (argument in radians)
TAN()	Tangent (argument in radians)
ATAN()	Arctangent (answer in radians)
NORM()	Standard normal density
CNORM()	Standard normal cumulative distribution function
CNORMI()	Inverse of the standard normal cumulative distribution function
LNORM()	Log of normal density
LCNORM()	Log of cumulative normal
DLCNORM()	Derivative of LCNORM = inverse Mills ratio
GAMFN()	Gamma function (not Gamma density)
LGAMFN ()	Log of Gamma function
DLGAMFN()	Derivative of LGAMFN = DIGAMMA()
TRIGAMMA()	Derivative of DIGAMMA() [non-differentiable]
FACT()	Factorial: FACT(X) = X! = GAMFN(X+1)
LFACT()	Log of factorial
SIGN()	Sign: -1 for X<0, 0 for X=0, 1 for X>0 [deriv=0]
POS()	Positive: POS(X) = max(0,X). Note: "MIN(A,B)" = B - POS(B-A) "MAX(A,B)" = A + POS(B-A)
MISS()	Missing: 1 for X missing, 0 otherwise [non-differentiable]
INT()	Truncate to an integer (round towards 0) [non-differentiable]
CEIL()	Ceiling (round away from 0) [non-differentiable]
ROUND()	Round to nearest integer (.5 rounds to 1) [non-differentiable]

[29] For matrix functions see the MATRIX command.

Relational and logical operators are the following

Operator	**.OP.**	**Description**
=	.EQ.	gives the value 1 when the variables on the left and on the right are equal; otherwise it is zero.
^= or ~=	.NE.	gives the value 1 when the variables on the left and on the right are not equal; otherwise it is zero.
<	.LT.	gives the value 1 when the variable on the left is less than the variable on the right; otherwise it is zero.
>	.GT.	gives the value 1 when the variable on the left is greater than the variable on the right; otherwise it is zero.
<=	.LE.	gives the value 1 when the variable on the left is less than or equal to the variable on the right; otherwise it is zero.
>=	.GE.	gives the value 1 when the variable on the left is greater than or equal to the variable on the right; otherwise it is zero.
&	.AND.	gives the value 1 when both the variable on the left and on the right are positive.
|	.OR.	gives the value 1 when either the variable on the left or the variable on the right is positive.
^ or ~	.NOT.	gives the value 1 when the variable on the right is negative or zero

Note that the .OP. form of the relational and logical operators is the alternative to the symbolic notation (but it cannot be used in nested DOT loops).

As many parentheses as necessary may be used to indicate the order of evaluation of a formula. Parentheses [] and { } are treated as (). In the absence of parentheses, evaluation proceeds from left to right in the following order:

1	Functions
2	Exponentiation (**)
3	Multiplication and division
4	Addition, subtraction, and negation (unary -)
5	Relational operators
6	.NOT. (^)
7	.AND. (&) and .OR. (|)

A.5 Rules for composing TSP statements:

Every statement begins with a command name. Exceptions:

```
X=Y;                  ? implicit GENR
X(I)=Y(I);            ? implicit SET
100 <statement>;      ? statement label for GOTO
```

The command name may be abbreviated, as long as it is uniquely identified.

Many statements can have options specified in parentheses after the command name. Option names may be abbreviated, like command names. There are three kinds of options:

Boolean options, either on or off. On is specified by the name of the option, as in PRINT, and off is specified by the option name with NO in front of it, as in NOPRINT.

Options of the form option name = option value. The value may be the name of a variable, a numerical value, or just a keyword, depending on the context.

Options which give lists of variables, and are of the form option name = (list of variables). Note that the parentheses are required, unless the list contains only one name, or the list is a listname.

A few commands can be followed by an algebraic formula: GENR, SET, SMPLIF, SELECT, FRML, IDENT, IF, GOTO.

Most commands are followed by one or more series names, separated by commas or spaces. These series names may include lags. An implicit list (such as X1-X5) can be used directly in a statement without making an intermediate listname. See the LIST command for a complete description of implicit list syntax.

The end of a statement is marked by a semicolon (;) or dollar sign ($).

A.6 Character Set for TSP

Character	Symbol	Use
letter	A to Z,_#% @	Parts of names. Lowercase letters are allowed on most computers; they are treated like uppercase letters. # % cannot be used for matrix names.
digit	0 to 9	Parts of numbers or names.
decimal point	.	Marks the decimal point in numbers; sets off logical operators; specifies string substitution in the DOT procedure.
comma	,	Separates the words in a list; spaces may be used, but commas are often preferred for clarity.
colon	:	Part of date.
semicolon	;	Marks the end of a statement.
dollar sign	$	Equivalent to ; . Semicolon is preferred.
quotation mark	"	Marks the beginning and end of a text string (title or filename); specifies matrix inversion.
apostrophe	’	Marks a text string; specifies matrix transposition.
parentheses	() [] {}	Encloses a list of options or expressions/lags in algebraic formulas.
question mark	?	Delimits the beginning of comments. (Comments are terminated by the end of the input line or logical record.)
plus sign	+	Specifies addition.
minus sign	-	Specifies subtraction, a lag, or a list.
star	*	Specifies multiplication or is part of power (**).
slash	/	Specifies division.
pound sign	#	Matrix Kronecker product (⊗).
percent	%	Matrix Hadamard product (element by element).
equal sign	=	Specifies equality or definition of data; relational operator (.EQ., .NE., .LE.,.GE.).
ampersand	&	Logical operator (.AND.).
vertical bar	|	Logical operator (.OR.).
caret or hat	^	Logical operator (.NOT., .NE.) or power (**), depending on context.
tilde	~	Logical operator (.NOT., .NE.).
less than	<	Relational operator (.LT., .LE.).
greater than	>	Relational operator (.GT., .GE.).
continuation	\	Continuation of a line (interactive).
miscellaneous	!`	Reserved for future use.

B. DIFFERENCES BETWEEN TSP 5.0 AND EARLIER VERSIONS

This appendix lists the features added to TSP since the release of 4.5 in June 1999. Some of them are already available in more recent versions of TSP 4.5. These features include changes that apply to the entire program, new commands, and enhancements to specific commands. The last section lists the incompatibilities with earlier versions.

B.1 Syntax and general enhancements:

- dashed lists
- long names
- faster SORT

B.2 Printing and graphics enhancements:

- better formatted PRINT output; use of high-res graphics for regression plots
- new graphics options for density curves on histograms
- better treatment of missing and panel data in PLOT and GRAPH
- 3-dimensional graphing in GRAPH

B.3 New or substantially improved procedures

- many new panel data features (random and fixed effects estimation for PROBIT and AR1, fixed effects estimation for linear LSQ, 3SLS, GMM, FIML, one and two-way ML random effects for PANEL)
- block diagonal HS and autocorrelation consistent standard errors for panel data
- Interval regression procedure
- Kernel density or regression estimation
- Censored quantile regression estimation with bootstrap standard errors
- ANALYZ for series, to compute s.e.s on quantities that vary across observations
- new nonlinear second derivative approximations (numeric second derivatives, based on numeric or analytic first derivatives), new tuning parameters
- Common factor test in AR1; root moduli and extended sample ACF for Box-Jenkins routines

- Different instruments for different equations in GMM (easier specification)
- Finite sample standard errors for LIML; more diagnostic testing

B.4 Changes to other procedures

Too numerous to list. See the online document *New and Improved Features in TSP 5.0* and the *TSP 5.0 Reference Manual.*

B.5 Upward Incompatibilities

None noted as yet.

C. USING TSP ON A WINDOWS PERSONAL COMPUTER

Installation of TSP on your personal computer is described in a memo accompanying the CD ROM or diskettes. This appendix assumes TSP is successfully installed on your hard disk.

C.1 Windows Products Available

The products listed below are included on the installation CD ROM. **TSP** and **TLG** (Through the Looking Glass) are included in the basic price. The **TSP/GiveWin** option is available at an additional cost, and requires a special license code for installation for that reason. However, if you already have GiveWin, this version of TSP will work with it and there is no need to purchase a second GiveWin license (in this case, you may need to contact us for a license number in order to use TSP with Givewin).

TSP/GiveWin (tspgw2.exe and tsprun.dll)

Windows 32-bit native program, running simultaneously with GiveWin 2.X as the server. Runs on Windows 95/98/NT/2000/XP. Graphics in separate window(s), printable with any installed printer. Interactive and batch text output in a scrollable window. No memory limitations. Large arrays.

Win32 TSP (tsp.exe)

Windows 32-bit native program (runs on Windows 95/98/NT/2000/XP). No graphics at present. No memory limitations; included mostly for this reason. It also has larger arrays than DOS/Win TSP, so it can hold 20,000 variables and 30,000 arguments in a command or combined in equations.

DOS/Win TSP (tspdw.exe)

Extended DOS (DPMI) program (runs on DOS, Windows 3.x/95/98/NT, OS/2, but not on Windows 2000/XP). Graphics: DOS full screen; printable on HP Laserjet and PostScript printers. Memory is limited to 64MB VM on Windows 95/98. Limited to 16MB RAM on Windows NT, but the swap/paging file on NT can be set to 4-5 times RAM. One user has reported allocating 1600MB on Windows NT 4.0 on a PC with 256MB RAM and an enlarged swap file

C.1.1 Auxiliary Programs

The following two products are available at no cost on the TSP installation CD-ROM.

TSP Through the Looking Glass (TLG, tlg.exe)

Windows 32-bit native program. Interfaces to the DOS/Win and Win32 versions of TSP for Windows. Provides side-by-side text edit windows for TSP batch input and output files, and some editing commands customized for TSP. Click on the "TSP" button to refresh the output window after changing the input commands.

TSP Help File (tsp.chm)

Windows type help file, with full (indexed) contents of the TSP Reference Manual. This file is accessible through TSP/GiveWin or directly from Windows.

C.2 Using TSP/GiveWin

GiveWin is a special shell or "server" program which runs simultaneously with one or more "client" (or "module") programs like TSP (actually ***tspgw.exe*** and ***tsprun.dll***). It provides a great array of Windows interface features, while using the full command language of TSP.

C.2.1 Batch mode

The easiest way to run batch programs in Givewin is the following:

- Double-click on a .tsp file to start **GiveWin** and open the file.

or

- Double-click on **GiveWin** to start it (if it is not already active)
- Click on the **Open** toolbar button (icon: opening folder)

- Use the dialog box to select an input file (the last one you used is remembered). Also, the File menu keeps a list of the last 8 files that have been opened, so it is a useful way of opening a recently used file. GiveWin sets your working directory to match the input file, so data or input files in the same directory can be read without requiring full pathnames. Use the .tsp extension for all TSP source/batch files.

Then

- The file is opened in a window, where you can make changes if necessary, save it, etc. Press F1 when the cursor is on a word for context-sensitive help.
- Click on the **Run** toolbar button (icon: running stick figure)
- This puts the output into a scrolling text window which is updated in real time. It is great for estimating large or nonlinear models, since you can monitor their progress, or scroll up to look at previous results while waiting for them to finish.
- Click on the **[x]** button on the output window to (optionally) save and close the output file.
- Click back to the input window, modify some commands, and click on the Run button to repeat the process.

During the install process, the file type .tsp is associated with GiveWin. When you are viewing a list of filenames in a folder (when starting from "My Computer" or Explorer), you will see TSP icons for these files, and you can right-click on the filename, and choose to Run it or Open it with GiveWin.

Note that GiveWin can only run one TSP batch file at a time, although you can have several open for editing at the same time. If you need to run two or more programs simultaneously, or in a fixed sequence, use DOS/Win TSP or Win32 TSP (see section D.2).

C.2.2 Interactive mode

Interactive use of TSP is easily done from GiveWin:

1. Double-click on **GiveWin** to start it (if it is not already active)
2. From the GiveWin menu, choose **Modules/Start TSP**
3. If necessary, choose a Working folder for the interactive session (it remembers the previous one)
4. Click on **Start TSP Session** (or press Enter)
5. Type your commands at the prompt, as usual. Press F1 when the cursor is

on a word for context-sensitive help. Use the arrow keys to recall/edit/rerun previous commands. Output is to the same scrolling text window, so you can scroll back to review or copy/paste previous results.

C.2.3 Filenames

All Windows filenames are supported directly in TSP/Givewin - short, long, containing blanks, etc.

C.2.4 Graphics

The PLOT and GRAPH commands automatically produce high resolution graphics in separate windows (the PREVIEW option is always on by default, in interactive and batch modes). The graphs can be enlarged, resized, printed, saved, or closed using the usual window controls. They can also be edited: change the line type, add plotting symbols, add text, etc. Just double-click in the TSP Graphics window, to bring up the Graphics Properties dialog box. See the chapter in the GiveWin manual for more information on editing graphs. The easiest way to print is to use the Print toolbar button.

Several graph windows can be created at the same time, although it may be tedious to cycle between them. In interactive mode, each graph goes to a window labelled with the command name (replacing any previous graph generated by that command). In batch mode, successive graphing commands create separate windows labelled with the type of graph and a sequence number, so the graphs can be viewed in any order or placed side by side. For example, each PLOT command creastes separate windows entitled “Plot 1, Plot 2, etc.). The WINDOW="name" option on the PLOT or GRAPH command can be used to name the graphics windows differently, or to overwrite older graphs with newer ones. If you are having trouble managing a lot of graphics windows, use the Window menu in GiveWin to tile or cascade the windows, or to select them by name.

If you have many graphics windows and want to exit GiveWin, just close GiveWin and all the graphics windows will be closed. If you want to save any of the graphics, use the Givewin Save file command (from the File menu) with the relevant graphics window on top. Graphics may be saved as postscript filew, enhanced metafiles (useful for inserting into documents), or in Givewin’s graphics format.

C.2.5 Other special features

GiveWin databases: Use the File/Open menu in GiveWin to open many types of database files (spreadsheets, **.in7** (GiveWin) files, etc. -- essentially any file which contains the variable names with the data). Data can be viewed, graphed, edited, etc. -- see the GiveWin manual. If you have GiveWin databases open and you start an interactive TSP session, you will be asked if you want to read each database into TSP. This is convenient for loading data you are already working with in GiveWin, and it also provides a way of accessing variables in native GiveWin (**.in7**) files which TSP does not read directly. You can also open databases from GiveWin while an interactive TSP session is active. Use the RELOAD; command (special to TSP/GiveWin only) to read them into your TSP session. GiveWin databases cannot be read directly by name (or in batch mode) into TSP, unless they are file types like .WKS or Excel. For files that you need to access in batch mode or repeatedly in interactive mode, it may be useful to open them in GiveWin, and then save them in a format like **.wks** that TSP supports directly.

CD command: the CD command can be used to show or change the current directory. It is useful in locating input or data files not in the current working folder. Usually it is best to locate all such files in the same folder for a given project, and then use the opening dialog box (see D.4.2) to select that as your current folder. CD with no arguments shows your current directory. CD with an argument changes your current directory. The argument does not have to be in quotes; they are added automatically if necessary; for example,

```
cd ..
```

works fine.

Known Quirks (Bugs?) - For unknown reasons, the DIR and SYSTEM commands do not send any output to the output window under Windows NT. They produce normal output under Windows 95 and 98.

C.3 Using DOS/Win TSP or Win 32 TSP

These two programs can be run in exactly the same way – either from **TLG** or from the "Command Prompt", see sections D.2.1 and D.2.2 below.

Win 32 TSP can handle long filenames and filenames with imbedded blanks. It also contains an internal CD command, which can be used to view or change your current directory, if you are in interactive mode and need to access several input or data files in a different directory. (In batch mode, we recommend using a full quoted pathname for the filename that includes all the relevant directory names). In this version of TSP, the arrow keys have no effect in interactive mode, and there are no graphics available (see **TSP/Givewin** for graphics in 32-bit Windows mode).

C.3.1 Batch mode

Batch mode involves typing your TSP commands into a file, and then running them with TSP to create an output file. It is the best way to do production work with TSP, because the commands can be read and changed easily, and they can be rerun at any time to reproduce or update results. Most experienced users will eventually migrate to this method, although interactive mode (see below) is valuable for learning or developing programs.

The easiest way to use TSP in batch mode is with Givewin (see section D.3 below) or with the **TLG** program (see D.1.4 above). In **TLG**, type your commands into the left hand window, and then click on the "TSP" button to see output in the right hand window. Press F1 when the cursor is on a word for context-sensitive help.

If you already have a favorite text editing program, you can use it instead of TLG. Type or modify your input file (containing TSP commands), and make sure it is saved as a "plain text" file. Suppose that your program is called `prog1.tsp`. Use the "Command Prompt" Windows program (sometimes called MS-DOS prompt) to issue the command

```
tsp prog1
```

Output from this TSP program will be in the file `prog1.out`, which you can then examine with your text editor. Arrow keys can be used to cycle between editing the input file, running TSP, and viewing the output file. To print some or all of the output file, use a fixed width font, such as Courier or Lineprinter. Built-in Windows programs such as WordPad can do this. Using the "Command Prompt" requires some knowledge of the DOS CD and PATH commands, so that you can run programs in a directory separate from the directory where TSP is installed. If you need to run several batch programs in sequence, you can put a series of TSP commands into a batch file (`.bat` file), and run it at the Command Prompt by typing the batch file name. For example, to run the file `run5.bat`, just type

```
run5
```

C.3.2 Interactive mode

Sometimes you don't know what commands to use ahead of time, because you need to look at the data or their internal documentation first. Or you may want to estimate a model or two before undertaking larger models. In these cases, it may be helpful to run TSP in interactive mode. Since the commands you type will be saved in a file called BKUP.TSP, you can also make a transition to batch mode later.

The easiest way to run TSP in interactive mode is with **Givewin**, but you can also do this with **DOS/Win** and **Win32 TSP**. Double-click on the TSP icon in TLG. Type your commands at the “?” prompt. Use the arrow keys to recall/edit/rerun previous commands. Output is "paged" so that it does not go off the screen too quickly.

If you need control of your current directory (to find data or input files), it is best to use the "Command Prompt" (MS-DOS Prompt). After you have used CD to set the desired directory, give the TSP command (click on TSP).

C.3.3 Filenames in Windows (DOS/Win TSP only)

DOS/Win TSP only understands original "DOS" filenames -- those with 8 characters or less in the directory and filename. If you have created some long directory names or filenames, or names with imbedded spaces, you will need to use the DOS "DIR" command to see what the corresponding "DOS" filenames are. For example, if you have a file called c:\Program Files\myfile.tsp, its "DOS" equivalent name will be something like c:\progra~1\myfile.tsp.

C.3.4 Graphics (DOS/Win TSP only)

The PLOT and GRAPH commands use the PREVIEW option to create full-screen graphics, so make sure the Properties/Screen/Usage of TSP.EXE is set to Full-screen (not Window). To check or set the Properties, start at the My Computer icon, and double-click down through C: , Program Files , and TSP 4.5 . Right-click on TSP.EXE and choose Properties and then Screen (this is the menu name on Windows NT; it may be slightly different on Windows 95/98). Only one graph can be on screen at a time; you need to press a key to restore the usual text output screen. Even though TSP takes over the full screen, you can still "switch tasks" to run other Windows programs -- use Ctrl-Esc to do this.

If you get a blank screen when you expect a graph, try the TSP command

```
OPTIONS DISPLAY=VGA;
```

to set your monitor type conservatively. If this works, you should insert this command into your LOGIN.TSP file. To print a graph, use the DEV= option; only a few printers such as HP LaserJet 3 (DEV=LJ3) and PostScript (DEV=PS) are supported, although DEV=LJ3 will produce correct output on most LaserJet printers.

Inserting graphics files into documents

DOS/Win TSP graphics can be placed in word processing files. Start by using the options

```
DEV=LJ3,FILE='name.HPG',PREVIEW
```

in your GRAPH or PLOT command. This will store the graph in the file name.HPG.

To place the graph in a MS Word document, open the Word file and choose Insert/Picture. In the dialog box, specify name.HPG and choose the filter "HP Graphics Language". (If the filter is not there, you will need to install it from your Word installation disks). Click OK and the graph will appear. You can then resize or edit it as you wish.

To place the graph in a WordPerfect document, you must first edit it as a plain DOS or ASCII file. On the first line, remove the first 3 or 4 characters so that it reads INSP1PA. Delete the last line, which will contain control characters such as ^[L, ^[E, etc. Save the file, and then open your document using WordPerfect. Choose Graphics/Figure from the menu. A file choice dialog box will appear, where you can specify name.HPG. Click on OK and the graph will appear; you can edit or resize it. Note: if you are using WordPerfect 5.1 or earlier, you will need to use the WordPerfect GRAPHCNV program to convert the HPG file to a WPG file before importing it into WordPerfect. You can also import PostScript (.PS) files into WordPerfect, but you cannot resize them.

C.4 Using TSP Through the Looking Glass (TLG)

TLG has already been described briefly above in sections D.2.1 and D.4.1. It also has its own Help file that describes its menus and its special editing features, which include:

1. autoformat input file (color comments green, etc.)
2. block indent or unindent
3. block comment or uncomment
4. search for "*** ERROR" in the output window

C.5 Using the Online Help System

The help system was described briefly in section D.1.5. To launch it from TSP/GiveWin, choose Modules/Start TSP Help. When typing or editing commands in GiveWin or TLG, press F1 when the cursor is on a word for context-sensitive help (this accesses tsp.chm). To launch it directly from Windows, double-click on tsp.chm.

C.6 Common problems

Lists of common installation and runtime problems can be found on our web page http://www.tspintl.com, under Support/FAQ.

By far the most common problem noted by users is opening data files -- "TSP says File Not Found, but I know it's there!!" Getting the computer to understand where you have put your data and how to read it appropriately can often be the most timeconsuming and frustrating part of a project. In TSP, you normally use the READ command to input your data, and you will probably need to specify a full pathname for the file on this command (although it is not case sensitive on the PC). A "full pathname" usually includes the full set of directory names, and the disk drive letter (especially if it is not C:).

In Windows, people often save data in spreadsheet files from their spreadsheet program, without knowing what the full set of directory names is, so this can occasionally be difficult to ascertain. One easy way to reveal the full pathname is to click on the Start button, and then Find/Files or Folders. Type in the filename under Named: (if you don't remember it, reopen your spreadsheet program), and click on Find Now (or press Enter). The search may take a 3-20 seconds, and should present you with a list of one or more locations for the file. The location is shown in "In Folder". If the directory name ends with ... , it is long, and you need to click and hold the vertical line between "In Folder" and "Size" (it becomes a "+" double arrow) -- drag it to the right, until there is enough room to show the full directory name. Use this name (with the filename) in quotes in your READ command. If you are using DOS/Win TSP and there are any long directory names, see section D.2.3 regarding the "DOS" version of the filename.

Some users save their files to a floppy disk (A:filename), to avoid these problems of deciphering the directory name. Another solution is to use your spreadsheet program to save the file in the same folder as your TSP programs, or move or copy it to that folder.

A related problem is that Windows often hides the "file types" (formerly known as "filename extensions" in DOS). These are a part of the full filename, and needed in TSP. Because Windows tends to hide them, a user who does not see the extension may type it in by hand when saving the file, and then the file will actually have the extension in twice. For example, if FOO is saved by Excel, the filename FOO.XLS is saved. If the user types FOO.XLS when saving in Excel, the filename FOO.XLS.XLS is sometimes created (depending on the version of Excel). So if Find/Files or Folders fails to locate FOO.XLS, try searching for FOO.XLS.XLS or *.XLS ! If FOO.XLS.XLS is found, it would be best to rename it, by going back into Excel and using Save As.

D. USING TSP ON THE APPLE MACINTOSH

Installation of TSP on your Mac is described in a memo accompanying the program diskettes. This appendix assumes that TSP has been successfully installed on your hard disk. As a reminder, note that the following equipment is required in order to run TSP:

- System 6-9 or X
- 2.5MB RAM (or more)
- hard disk (with at least 1.2Mb free)
- CD-ROM or 1.44M floppy disk drive (to install the program)
- PowerPC CPU

With the following exceptions, this is the same program as the TSP that runs on DOS/Win, VAX/VMS, unix, etc.. The Mac-specific features are:

- default memory sizing via the Get Info resource
- initial dialog box/file list to choose batch input file or interactive mode
- adjustable-sized output and Plot/Graph windows
- dialog boxes when printing graphics, to choose Portrait/Landscape mode interactively
- double-clicking on output file to invoke your editor (the same text editor that created the input file).

D.1 Running TSP in Interactive mode

Move into the TSP folder by double-clicking on it, and double click on the TSP icon to start TSP.

First, the screen clears and TSP 5.0 appears in the title bar of a scrollable output window. Second, a short copyright message for the Fortran compiler will appear, and a prompt will be given below:

```
Select batch input file [or hit Cancel for interactive]:
```

followed by a list of text files in your folder which can be selected for batch operation. Click on the Cancel box to run TSP interactively (or see below for batch operation). Finally, the TSP version and address will be displayed, and you will be prompted for TSP commands:

```
1 ?
```

The text is in 9 point Monaco (a non-proportional font). There is a blinking cursor bar indicating where your commands are typed. An I-beam (text) cursor is also initially displayed. You can type commands at this time, and output will appear in the same window. Press RETURN to end a command (the ; is not required), or use \ and RETURN to continue a command onto further lines. Some simple commands to try out are HELP and SHOW, if you are running TSP for the first time. The END or QUIT command terminates TSP. Your commands during the interactive session are saved in a BKUP.TSP file, unless you exit with the QUIT command. This file can be renamed, edited and used to run TSP in batch mode.

You can select text in the output window and print it or paste it into files. You can also scroll the output window up and down to look at previous results from different commands. If you move the cursor around to do these things, you can hold down the Option (alt) key and press RETURN to return the cursor to the insertion point for typing more commands. Otherwise you will just get beeps when you try to type. You can also move the cursor there manually with the mouse - the insertion point is one space right of the lowest question mark.

D.2 Running TSP in Batch Mode

Follow the steps above, but select a text file containing TSP commands from the list of files (instead of clicking on Cancel). The convention is to name these files with an extension .TSP, such as KLEINLSQ.TSP. TSP output (regression results, etc.) is stored in a text file with the same name but with the extension .OUT, such as KLEINLSQ.OUT. TSP puts a message on the screen regarding the input file name. It also mentions that Command-. (Apple key plus period key) can be used to halt the program.

When TSP is finished, open the output file from your text editor to examine the results; you will probably want to use a non-proportional font such as Monaco so that the text is aligned properly for printing/viewing. The output file has the same creator as the input file, so you can just double-click on it to invoke your editor. You will probably prefer to task-switch to examine the output file and revise the input file; then you can rerun TSP on the same file without having to reload it into memory.

D.3 File formats and Names

Program and data files are normally plain text files with at most 80 characters per line. You can read free format data files of any width (i.e. much larger than 80). TSP will tolerate line feed and tab characters in files (they get translated into spaces), but you can remove them from ported DOS files by selecting the Text format in the MS-DOS-to-Mac menu of the Apple File Exchange.

For filenames, the same rules apply as on the PC; the maximum length is 128 characters. Any valid pathname is acceptable; the Macintosh uses : (colon) to separate devices and folders in a pathname (for example, HD:TSP:FOO.DAT is a valid pathname). Therefore, you may easily keep your input, output and data files in separate folders if you wish.

All files require a name (no names are automatically generated, except those displayed in square brackets in prompts). This means that the UNIT= option on READ, WRITE, or LOAD must not be used without the FILE= option unless the unit has previously been opened with a specific name attached. As always, if FILE= is used alone, TSP will generate a unit number for you.

D.4 Graphics in Mac TSP: PLOT, GRAPH

On the Mac, all printers are supported. The plot remains on the screen until you press a key or the mouse. You can press p, P or %P at this point to make a hard copy before the plot window is closed. Unless dash is being used, the WIDTH option is set by default when printing distinguish multiple lines on the hardcopy from color systems. The PLOT window size is determined from the TSP output window size, so if you click on its maximize button, you will get a larger graphics window. The hardcopy size is determined from the printer page dimensions (you can select portrait or landscape in the dialog box).

For general information on graphics in TSP, see GRAPH and PLOT in Chapter 6 and the *TSP Reference Manual.*

E. USING TSP ON A UNIX COMPUTER

There are 3 standard ways to run TSP from the unix command line:

1. tsp to run interactively
2. tsp foo to run in batch mode, input from foo.tsp , output to foo.out
3. tsp FOO to run in batch mode, input from FOO.TSP,
output to FOO.OUT

In the above, foo is just an example; other filenames and full pathnames will also work fine. The output file created above could be printed with page breaks by the unix command:

```
lpr foo.out
```

Of course, much time and paper can be saved by checking for errors in the output file first with a text editor or a unix command such as

```
fgrep "*** ERROR" foo.out
```

One aspect of TSP that may seem confusing under unix is that any unquoted filenames in commands are translated to uppercase. For example, `out bar;` creates the databank BAR.TLB. However, this is only confusing if one expects to find bar.tlb with the ls command. The in bar; command will still find BAR.TLB and read it with no problems. The READ command will look for files in both uppercase and lowercase.[30]

For information on reading and writing Excel and Lotus files see section 16.2.

The READ and WRITE commands support the ~username syntax in filenames, so that a filename can be specified relative to some user's HOME directory, like in the C-shell (csh). For example:

```
read(file='~joe/data/j5.dat') x y;
```

looks for the file j5.dat in the user joe's subdirectory named data.

[30] An implication of this is that because TSP, unlike UNIX, makes no distinction between upper and lower case names, you cannot store files named foo.tsp and FOO.TSP in the same directory and expect TSP to distinguish them.

```
read(file='~/world/finn.dat') y1-y4;
```

would look for the file finn.dat in your own subdirectory named world. Of course, it is not necessary to specify full pathnames like this if your data files are in your current directory.

F. REFERENCES

The list below is comprehensive: we have included all the references that appear either in this *User's Guide* or in the *Reference Manual/Online Help System.* We would appreciate it if you would bring any missing or incorrect references to our attention by emailing ***info@tspintl.com.***

Abramovitz, Milton and Irene A. Stegun. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, John Wiley & Sons, New York, 1972.

ACM, *Collected Algorithms*, New York, 1980.

Ahn, S.C., and P. Schmidt, "Efficient Estimation of Panel Data Models with Exogenous and Lagged Dependent Regressors," *Journal of Econometrics* 68 (1995) 5-27.

Ahrens, H., and R. Pincus, "On two measures of unbalancedness in a one-way model and their relation to efficiency," *Biometric Journal* 23 (1981), pp. 227-235.

Albert, A., and J.A. Anderson, "On the Existence of Maximum Likelihood Estimates in Logistic Regression Models," *Biometrika* 71 (1984).

Almon, Clopper, *Matrix Methods in Econometrics*, Addison-Wesley Publishing Company, Reading, Mass., 1967, pp. 115-120.

Almon, Shirley, "The Distributed Lag Between Capital Appropriations and Expenditures," *Econometrica* 33(1965), pp. 178-196.

Amemiya, Takeshi, *Advanced Econometrics*, Harvard University Press, Cambridge, Mass., 1985.

Amemiya, Takeshi, "Qualitative Response Models: A Survey*," Journal of Economic Literature* 19 (1981), pp. 1483-1536.

Amemiya, Takeshi, "Tobit Models: A Survey," *Journal of Econometrics* 24 (1981), pp. 3-61.

Amemiya, Takeshi, "The Maximum Likelihood and the Nonlinear Three-Stage Least Squares Estimator in the General Nonlinear Simultaneous Equation Model," *Econometrica* 45 (1977), pp. 955-975.

Amemiya, Takeshi, "The Nonlinear Two-Stage Least-Squares Estimator," *Journal of Econometrics* 2 (1974), pp. 105-110.

Anderson, T. W., N. Kunitomo, and K. Morimune, "Comparing Single Equation Estimators in a Simultaneous Equation System," Technical Report No. 1, Econometric Workshop, Stanford University, January 1985.

Andrews, Donald W. K., "Heteroskedasticity and Autocorrelation Consistent Covariance Matrix Estimation," *Econometrica* 59 (3), 1991, pp. 817-858.

Arellano, Manuel, and Stephen R. Bond, “Some Tests of Specification for Panel Data: Monte Carlo Evidence and an Application to Employment Equations,” *Review of Economic Studies* 58 (1991): 277-297.

Baltagi, Badi, *Econometric Analysis of Panel Data*, Wiley & Sons, New York, 1995 (first edition).

Baltagi, B. H. and Q. Li, "A Transformation That Will Circumvent the Problem of Autocorrelation in an Error-Component Model," *Journal of Econometrics* 48 (1991), pp. 385-393.

Bartlett, M.S., “The Statistical Significance of Canonical Correlations”, *Biometrika*, January 1941, pp. 29-37.

Barrodale, I., and F. D. K. Roberts, Algorithm #478, *Collected Algorithms from ACM* Volume II, Association for Computing Machinery, New York, 1980.

Beach, Charles M., and James G. MacKinnon, “A Maximum Likelihood Procedure for Regression with Autocorrelated Errors,” *Econometrica* 46(1978), pp. 51-58.

Bekker, P. A., “Alternative Approximations to the Distributions of Instrumental Variable Estimators,” *Econometrica* 63 (1994), pp. 657-681.

Belsley, David A., Kuh, Edwin, and Welsch, Roy E., *Regression Diagnostics: Identifying Influential Data and Sources of Collinearity*, John Wiley & Sons, New York, 1980, pp. 11-18.

Berndt, E. R., B. H. Hall, R. E. Hall, and J. A. Hausman, “Estimation and Inference in Nonlinear Structural Models,” *Annals of Economic and Social Measurement* 3(1974), pp. 653-665.

Berndt, E. R., and N.E. Savin, “Conflict Among Criteria for Testing Hypothesis in the Multivariate Linear Regression Model,” *Econometrica* 45(1977), pp. 1263-1278.

Bhargava, A., L. Franzini, and W. Narendanantham, “Serial Correlation and the Fixed Effects Model,” *Review of Economic Studies* XLIX (1982): 533-549.

Bilias, Y., S. Chen, and Z. Ying, “Simple Resampling Methods for Censored Regression Quantiles,” *Journal of Econometrics* 99 (2000), pp. 373-386.

Bishop, Y. M. M., S. E. Fienberg, and P. W. Holland, *Discrete Multivariate Analysis: Theory and Practice*, MIT Press, Cambridge, MA, 1975, pp. 486-502.

Bloom, David E., and Killingsworth, Mark R., “Correcting for Selection Bias Caused by a Latent Truncation Variable,” *Journal of Econometrics* 27(1985), pp. 131-135.

Blundell, Richard, and Stephen R. Bond, “Initial Conditions and Moment Restrictions in Dynamic Panel Data Models,” *Journal of Econometrics*. 87 (1998), pp. 115-143.

Blundell, Richard, and Richard J. Smith, "Initial Conditions and Efficient Estimation in Panel Data Models," University College London Discussion Paper in Economics No. 91-04, 1991. Published in French as "Conditions initiales et estimation efficace dans les modeles dynamiques sur donnees de panel: une application au comportement d'investissement des entreprises," *Annales d'Economie et de Statistique*, No. 20/21 (1991), pp. 109-123.

Blundell, Richard, Rachel Griffith, and Frank Windmiejer, "Individual Effects and Dynamics in Count Data Models," *Journal of Econometrics* 108 (2002), pp. 113-131.

Bollerslev, Tim, "Generalized Autoregressive Conditional Heteroskedasticity," *Journal of Econometrics* 31(1986), pp. 307-327.

Box, George P., and G.M. Jenkins, *Time Series Analysis: Forecasting and Control*, Holden-Day, New York, 1976.

Brown, Barry W., DCDFLIB. http://odin.mdacc.tmc.edu (downloaded v1.1, 4/1998).

Brown, R. L., J. Durbin, and J. M. Evans, "Techniques for Testing the Constancy of Regression Relationships over Time", *Journal of the Royal Statistical Society - B*, 1975.

Buse, A., "Efficient Estimation of a Structural Equation with First-Order Autocorrelation," *Journal of Quantitative Economics* 5(1989), pp. 59-72.

Calzolari, Giorgio, and Gabriele Fiorentini, "Alternative Covariance Estimators of the Standard Tobit Model," Paper presented at the World Congress of the Econometric Society, Barcelona, August 1990.

Calzolari, Giorgio, and Lorenzo Panattoni, "Alternate Estimators of FIML Covariance Matrix: A Monte Carlo Study," *Econometrica* 56 (1988), pp. 701-714.

Cameron, A. Colin, and Pravin K. Trivedi, "Count Models for Financial Data," Maddala and Rao (eds.), *Handbook of Statistics, Volume 14: Statistical Methods in Finance*, Elsevier/North-Holland, 1995.

Cameron, A. Colin, and Pravin K. Trivedi, *The Analysis of Count Data*, University of California at Davis and Indiana University, draft manuscript, 1996.

Cameron, A. C., and F. A. G. Windmeijer, "R-Squared Measures for Count Data Regression Models with Applications to Health Care Utilization," *Journal of Business and Economic Statistics* 14 (1996): 209-220.

Cameron, A. C., and F. A. G. Windmeijer, "An R-Squared Measure of Goodness of Fit for Some Common Nonlinear Regression Models," *Journal of Econometrics* 77 (1997): 329-342.

Campbell, John Y., and Pierre Perron, "Pitfalls and Opportunities: What Macroeconomists Should Know about Unit Roots", in Olivier Jean Blanchard and Stanley Fischer (eds.), *NBER Macroeconomics Annual*, MIT Press, Cambridge, Mass., 1991.

Census Bureau, *Seasonal Analysis of Economic Time Series*, proceedings of the Conference on the Seasonal Analysis of Economic Time Series, September 1976.

Chamberlain, Gary, “Multivariate Regression Models for Panel Data,” *Journal of Econometrics* 18 (1982), pp. 5-46.

Chamberlain, Gary, “Panel Data,” in Griliches and Intriligator (eds.), *The Handbook of Econometrics*, Volume II, North Holland Publishing Co., Amsterdam, 1985.

Cheung, Yin-Wong, and Lai, Kon S., “Lag Order and Critical Values of the Augmented Dickey-Fuller Test,” *Journal of Business & Economic Statistics* 13 (July 1995): 277-280.

Cochrane, D., and G. H. Orcutt, “Application of Least Squares Regression to Relationships Containing Autocorrelated Error Terms,” *JASA* 44(1949), pp. 32-61.

Cooley, T. F., and Edward Prescott, “Varying Parameter Regression: A Theory and Some Applications,” *Annals of Economic and Social Measurement* 2(1973), pp. 463-474.

Cooper, J. Phillip, “Asymptotic Covariance Matrix of Procedures for Linear Regression in the Presence of First-Order Autoregressive Disturbances,” *Econometrica* 40(1972), pp. 305-310.

Cooper, J. Phillip, “Time-Varying Regression Coefficients: A Mixed Estimation Approach and Operational Limitation of the General Markov Structure,” *Annals of Economic and Social Measurement* 2(1973), pp. 525-530.

Cooper, J. Phillip, “Two Approaches to Polynomial Distributed Lag Estimation: An Expository Note and Comment,” *The American Statistician*, June 1972, pp. 32-35.

Cragg, J. G., and S. G. Donald, "Testing Identifiability and Specification in Instrumental Variable Models," *Econometric Theory* 9 (1993), pp. 222-240.

Cummins, Clint, and Bronwyn H. Hall, *Time Series Processor Version 4.0/4.1 Programmer's Manual*, TSP International, Stanford, California, 1986.

Cushman, David O., Sang Sub Lee, and Thorsteinn Thorgeirsson, “Maximum Likelihood Estimation of Cointegration in Exchange Rate Models for Seven Inflationary OECD Countries,” *Journal of International Money and Finance*, June 1996.

Davidson, Russell, and James G. MacKinnon, *Estimation and Inference in Econometrics*, Oxford University Press, 1993.

Davis, Peter, “Estimating Multi-Way Error Components Models with Unbalanced Data Structures,” *Journal of Econometrics* 106 (July 2002), pp. 67-95.

Dickey, D.A., and W.A. Fuller, “Distribution of the Estimators for Autoregressive Time Series with a Unit Root,*” Journal of the American Statistical Association* 74 (1979): 427-431.

DiDinato, A.R. and Morris, Alfred H. Jr., “Computation of the Incomplete Gamma Function Ratios and Their Inverse,” *ACM Transactions on Mathematical Software* 12, 1986, pp. 377-393.

DiDinato, A.R. and Morris, Alfred H. Jr., “Algorithm 708: Significant Digit Computation of the Incomplete Beta Function Ratios,” *ACM Transactions on Mathematical Software* 18, 1993, pp. 360-373.

Diewert, Erwin, “Exact and Superlative Index Numbers,” *Journal of Econometrics* 4(1976), pp. 115-146.

Divisia, F., *Economique rationnelle*, Gaston Doin, Paris, 1928.

Divisia, F., “L'indice monetaire et la theorie de la monnaie,” *Revue d'Economie Politique* 39(1925), pp. 842-861, 980-1008, 1121-1151.

Dufour, J-M, Gaudry, M. J. I., and Liem, T. C., “The Cochrane-Orcutt Procedure: Numerical Examples of Multiple Admissible Minima,” *Economics Letters* 6 (1980), pp. 43-48.

Durbin, J., “Boundary-crossing probabilities for the Brownian motion and Poisson processes and techniques for computing the power of the Kolmogorov-Smirnov test,” *Journal of Applied Probability* 8 (1971), pp. 431-453.

Durbin, J., “Testing for Serial Correlation in Least Squares Regression When Some of the Regressors are Lagged Dependent Variables,” *Econometrica* 38 (1970), pp. 410-421.

Durbin, J., “Tests for Serial Correlation in Regression Analysis Based on the Periodogram of Least Squares Residuals,” *Biometrika*, 1969.

Durbin, J., and G.S. Watson, “Testing for Serial Correlation in Least Squares Regression,” *Biometrika* 38 (1951), pp. 159-177.

Edgerton, David and Curt Wells, “On the Use of the CUSUMSQ Statistic in Medium Sized Samples”, *Oxford Bulletin of Economics and Statistics*, 1994.

Efron, Bradley, “Bootstrap Methods: Another Look at the Jackknife,” *Annals of Statistics* 7 (1979), pp. 1-26.

Efron, Bradley, *The Bootstrap, the Jackknife and Other Resampling Plans*, Philadelphia: SIAM, 1982.

Efron, Bradley, and G. Gong, “A Leisurely Look at the Bootstrap, Jackknife, and Cross-validation,” *American Statistician*, February 1983, 37(1), pp. 36-48.

Engle, Robert F., “Autoregressive Conditional Heteroskedasticity with Estimates of the Variance of U. K. Inflation,” *Econometrica* 50(1982), pp. 987-1008.

Engle, Robert F., “A General Approach to Lagrange Multiplier Model Diagnostics,” *Journal of Econometrics* 20(1982), pp. 83-104.

Engle, Robert F., “Wald, Likelihood Ratio and Lagrange Multiplier Tests in Econometrics,” in Griliches and Intriligator (eds.), *The Handbook of Econometrics*, North Holland Publishing Co., Amsterdam, 1985, pp. 776-826.

Engle, Robert F., and Clive Granger, “Cointegration and Error Correction: Representation, Estimation, and Testing,” *Econometrica* 55(1987), pp. 251-276.

Engle, Robert F., David M. Lilien, and Russell P. Robins, “Estimating Time Varying Risk Premia in the Term Structure: The ARCH-M Model,” *Econometrica* 55(1987), pp. 391-407.

Estrella, Arturo, “A New Measure of Fit for Equations with Dichotomous Dependent Variables,” *Journal of Business and Economic Statistics*, April 1998, pp. 198-205.

Faddeev, V. N., *Computational Methods of Linear Algebra*, (trans. C. Benster), Dover, New York, 1959.

Fair, Ray C., “The Estimation of Simultaneous Equation Models with Lagged Endogenous Variables and First-Order Serially Correlated Errors,” *Econometrica* 38(1970), pp. 507-516.

Fair, Ray C., *Specification, Estimation and Analysis of Macroeconomic Models*, Harvard University Press, Cambridge, MA, 1984.

Farebrother, R. W., “Algorithm AS 256”, *Applied Statistics* 39, 1990. Pascal code posted on StatLib. http://lib.stat.cmu.edu/apstat/

Fiorentini, Gabriele, Calzolari, Giorgio, and Panattoni, Lorenzo, “Analytic Derivatives and the Computation of GARCH Estimates,” *Journal of Applied Econometrics* 11 (1996), pp.399-417.

Fishman, George S., and Louis R. Moore, “A Statistical Evaluation of Multiplicative Congruential Random Number Generators with Modulus 231-1,” *JASA* 77 (1982), pp. 129-136.

Fitzenberger, Bernd, “A Guide to Censored Quantile Regressions,” in G. S. Maddala and C. R. Rao (eds.), *Handbook of Statistics*, Volume 15: *Robust Inference*, 1997, pp. 405-437.Fletcher, R., and M. J. D. Powell, “A Rapidly Converging Descent Method for Minimization,” *Comput. J*, Vol. 6, pp. 163-168.

Fuller, Wayne A., “Some Properties of a Modification of the Limited Information Estimator,” *Econometrica* 45: 939-953.

Gallant, A. Ronald, *Nonlinear Statistical Models*, Wiley, New York, 1987.

Gallant, A. Ronald, and Dale Jorgenson, “Statistical Inference for a System of Simultaneous, Non-linear, Implicit Equations in the Context of Instrumental Variable Estimation”, *Journal of Econometrics* 11 (1979), pp. 275-302.

Gallant, A. Ronald, and Alberto Holly, “Statistical Inference in an Implicit, Nonlinear, Simultaneous Equation Model in the Context of Maximum Likelihood Estimation”, *Econometrica* 48 (1980), pp. 697-720.

Geweke, John F., and Richard Meese, “Estimating Regression Models of Finite but Unknown Order,” *International Economic Review* 22 (1981), pp. 55-70.

Gill, Philip E., Walter Murray, and Margaret H. Wright, *Practical Optimization*, Academic Press, New York, 1981.

Gilli, Manfred, “Causal Ordering and Beyond,” *International Economic Review*, November 1992, pp. 957-971.

Gilli, Manfred, “Graph-theory based tools in the practice of macroeconometric modeling,” in S. K. Kuipers, L. Schoonbeek, and E. Sterken (eds), *Methods and Applications of Economic Dynamics*, North Holland, Amsterdam.

Godfrey, L. G., *Misspecification Tests in Econometrics*, Econometric Society Monograph, Cambridge University Press, Cambridge, England, 1988, pp. 143-145.

Goldfeld. S. M. and R. E. Quandt, *Nonlinear Methods in Econometrics*, North-Holland, Amsterdam, 1972.

Gourieroux, Christian, Alain Montfort, and Alain Trognon, “Pseudo Maximum Likelihood Methods: Theory,” *Econometrica* 52(1984), pp. 681-700.

Gourieroux, Christian, Alain Montfort, and Alain Trognon. “Pseudo Maximum Likelihood Methods: Applications to Poisson Models,” *Econometrica* 52 (1984): 701-720.

Greene, William H., “On the Asymptotic Bias of the Ordinary Least Squares Estimator of the Tobit Model,” *Econometrica* 49 (1981), pp. 505-513.

Gregory, Allan W., “Testing for Cointegration in Linear Quadratic Models,” *Journal of Business and Economic Statistics*, July 1994, pp. 347-360.

Griffiths, W. E., R. C. Hill, and P. J. Pope, “Small Sample Properties of Probit Model Estimators,” *JASA* 82(1987), pp. 929-937.

Griliches, Zvi, and J.A. Hausman, “Errors in Variables in Panel Data,” *Journal of Econometrics* 31(1986), pp. 93-118.

Griliches, Zvi, J. A. Hausman, and Bronwyn H. Hall, “Missing Data and Self Selection in Large Panels,” *Annals de l'INSEE* 30-31(1978), pp. 137-176.

Griliches, Zvi, and Michael D. Intriligator, *Handbook of Econometrics Volumes I, II, III*, North Holland, 1984, 1985, 1986.

Haerdle, W., *Applied Nonparametric Regression*, Cambridge University Press, Cambridge, 1990.

Hall, Bronwyn H., “The Effect of Takeover Activity on Corporate Research and Development,” in Auerbach, Alan (ed.), *Corporate Takeovers: Causes and Consequences*, 1988, pp. 69-96.

Hall, Bronwyn H., “The Relationship between Firm Size and Firm Growth in the U.S. Manufacturing Sector,” *Journal of Industrial Economics* 36(1987), pp. 583-606.

Hall, Bronwyn H., Zvi Griliches, and Jerry A. Hausman, “Patents and R&D: Is There a Lag?”, *International Economic Review* 27(1986), pp.265-283.

Hall, Bronwyn H., and Clint Cummins, *TSP Version 5.0 Reference Manual*, TSP International, Stanford, California, 2004.

Hall, Bronwyn H., “Estimation of the Probability of Acquisition in an Equilibrium Setting,” University of California at Berkeley IBER Working Paper No. 8887, 1987.

Hall, Robert E., “Polynomial Distributed Lags,” Econometrics Working paper No. 7, Department of Economics, MIT, July 1967.

Hall, Robert E., “Stochastic Implications of the Life-Cycle Permanent Income Hypothesis: Theory and Evidence,” *Journal of Political Economy* 86(1978), pp. 971-987.

Hanoch, Giora, “A Multivariate Model of Labor Supply: Methodology for Estimation,” in J. P. Smith (ed.), *Female Labor Supply: Theory and Estimation*, Princeton University Press, Princeton, 1980.

Hansen, C., J. A. Hausman, and W. Newey, “Weak Instruments, Many Instruments, and Microeconometric Practice,” MIT, Cambridge, Mass: working paper, 2004.

Hansen, Lars Peter, “Large Sample Properties of Generalized Method of Moments Estimators,” *Econometrica* 50(1982), pp. 1029-1054.

Hansen, Lars Peter, and Kenneth J. Singleton, “Generalized Instrumental Variables Estimation of Nonlinear Rational Expectations Models,” *Econometrica* 50(1982), pp. 1269-1286.

Harman, Harry H., *Modern Factor Analysis,* University of Chicago Press, First Edition (1960), Sec. 9.3 or Third Edition (1976), Sec. 8.3.

Harvey, Andrew C., *The Econometric Analysis of Time Series*, Cambridge: The MIT Press, third printing, 1993.

Harvey, Andrew C., *Forecasting, Structural Time Series Models, and the Kalman Filter*, Cambridge, Cambridge University Press, fifth printing, 1994.

Harvey, Andrew C., *Time Series Models*, 1981, Philip Allen, London.

Hausman, Jerry A., “Specification Tests in Econometrics,” *Econometrica* 46 (1978), pp. 1251-1272.

Hausman, Jerry A., Bronwyn H. Hall, and Zvi Griliches, “Econometric Models for Count Data with an Application to the Patents-R&D Relationship,” *Econometrica* 52(1984), pp. 909-938.

Hausman, Jerry A., and Daniel McFadden, “Specification Tests for the Multinomial Logit Model,” *Econometrica* 52 (1984): 1219-1240.

Heckman, James J., “Sample Selection Bias as a Specification Error,” *Econometrica* 47(1974), pp. 153-162.

Hildreth, C., and J. Y. Lu, “Demand Relations with Autocorrelated Disturbances,” *Research Bulletin* 276, Michigan State University Agricultural Experiment Station, 1960.

Hsiao, Cheng, *Analysis of Panel Data*, Cambridge University Press, Cambridge, England, 1986.

Imhoff, P.J., “Computing the Distribution of Quadratic Forms in Normal Variables,” *Biometrika* 48 (1961), pp. 419-426.

IMSL Library Reference Manual, IMSL, Inc., Houston, Texas, 1982.

Jarque, Carlos M., and Anil K. Bera, “A Test for Normality of Observations and Regression Residuals,” *International Statistical Review* 55 (1987): 163-172.

Jayatissa, W. A., “Tests of Equality Between Sets of Coefficients in Linear Regressions when Disturbance Variances are Unequal,” *Econometrica* 45 (1977), pp. 1291-1292.

Johansen, Søren, and Katarina Juselius, “Maximum Likelihood Estimation and Inference on Cointegration -- with Applications to the Demand for Money”, *Oxford Bulletin of Economics and Statistics*, 1990, p.169-210.

Jorgenson, Dale W., and Jean-Jacques Laffont, “Efficient Estimation of Nonlinear Simultaneous Equations with Additive Disturbances,” *Annals of Economic and Social Measurement* 4(1974), pp. 615-640.

Jorgenson, Dale, and Zvi Griliches, “Divisia Index Numbers and Productivity Measurement,” *Review of Income and Wealth*, Vol. 17(2), June 1971, pp. 227-229.

Judge, George, R. Carter Hill, William E. Griffiths, Helmut Lutkepohl, and Tsoung-Chao Lee. *Introduction to the Theory and Practice of Econometrics*, John Wiley & Sons, New York, 1988 (second edition).

Judge, George, et al, *The Theory and Practice of Econometrics*, John Wiley & Sons, New York, 1981, pp. 531-533.

Kalman, R. E., “A New Approach to Linear Filtering and Prediction Problems,” *Journal of Basic Engineering, Transactions ASME*, Series D 82 (1960): 35-45.

Keane, Michael P., and David E. Runkle, “On the Estimation of Panel-Data Models with Serial Correlation When Instruments are not strictly Exogenous,” *Journal of Business and Economic Statistics* 10(1992), pp. 1-29.

Klein, L. R., “Estimation of Interdependent Systems in Macro-Economics,” *Econometrica* 37 (1969): 171-192.

Knuth, Donald E., *The Art of Computer Programming*, Volume 2: Seminumerical Algorithms, Addison-Wesley, Reading, Mass., 1969.

Koenker, R. W., and G. W. Bassett, “Regression Quantiles,” *Econometrica* 46 (1978), pp. 33-50.

Krasker, William S., Kuh, Edwin, and Welsch, Roy E., "Estimation for Dirty Data and Flawed Models," Griliches and Intrilligator (eds.), *Handbook of Econometrics*, Volume I, North-Holland Publishing Co., New York, 1983, pp. 660-664.

Lancaster, Tony, *The Economic Analysis of Transition Data*, Cambridge: Cambridge University Press, 1990.

Leamer, Edward E*., Specification Searches: Ad Hoc Inference with Nonexperimental Data*, Wiley, New York, 1978, p. 114.

L'Ecuyer, Pierre, "Good Parameter Sets for Combined Multiple Recursive Random Number Generators," Operations Research 47, 1999. Available at http://www.iro.umontreal.ca/~lecuyer/papers.html

L'Ecuyer, Pierre, "Random Numbers for Simulation," *Communications of the ACM*, October 1990, pp. 85-97.

Ljung, G. M., and G. P. Box, "On a Measure of Lack of Fit in Time Series Models," *Biometrika* 66(1978), pp. 297-303.

Longley, James W., "An Appraisal of Least Squares Programs for the Electronic Computer from the Point of View of the User," *JASA* 62(1967), pp. 818-841.

Machado, J. A. F., and J. M. C. Santos-Silva, "Glejser's Test Revisited," *Journal of Econometrics* 97 (2000): 189-202.

MacKinnon, J. G., "Approximate Asymptotic Distribution Functions for Unit-Root and Cointegration Tests," *Journal of Business and Economic Statistics* (April 1994): 167-176.

MacKinnon, J.G., "Critical Values for Cointegration Tests," in *Long-Run Economic Relationships: Readings in Cointegration*, eds. R.F. Engle and C.W.J. Granger, New York: Oxford University Press, (1991): 266-276.

MacKinnon, James G., and Halbert White, "Some Heteroskedasticity Consistent Covariance Matrix Estimators With Improved Finite Sample Properties," *Journal of Econometrics* 29, pp.305-325.

Macurdy, Thomas E., "Asymptotic Properties of Quasi-Maximum Likelihood Estimators and Test Statistics," NBER Technical Working Paper No. 14, 1981.

Macurdy, Thomas E., "An Empirical Model of Labor Supply in a Life Cycle Setting," *Journal of Political Economy* 89(1981), pp. 1059-1085.

Macurdy, Thomas E., "A Guide to Applying Time Series Models to Panel Data," Manuscript, Stanford University, 1985.

Macurdy, Thomas E., "The Use of Time Series Processes to Model the Error Structure of Earnings in a Longitudinal Data Analysis," *Journal of Econometrics* 18(1981), pp. 83-114.

Maddala, G. S., *Introduction to Econometrics*, Macmillan, New York, 1988.

Maddala, G. S., *Limited-dependent and Qualitative Dependent Variables in Econometrics*, Cambridge University Press, New York, 1983.

Maddala, G. S., *Econometrics*, McGraw Hill Book Company, New York, 1977.

Maddala, G. S., "The Use of Variance Component Models in Pooling Cross Section and Time Series Data," Econometrica 39(1971), pp. 341-58.

Manski, Charles, and Daniel McFadden, Structural Analysis of Discrete Data with *Econometric Applications*, MIT Press, Cambridge, Mass., 1983.

McCurdy, Thomas H., and Ieuan G. Morgan, "Testing the Martingale Hypothesis in Deutsche Mark Futures with Models Specifying the Form of Heteroskedasticity," *Journal of Applied Econometrics* 3(1988), pp. 187-202.

McFadden, Daniel, "Conditional Logit Analysis of Qualitative Choice Behavior," in P. Zarembka (ed.), *Frontiers in Econometrics*, Academic Press, New York, 1973.

McFadden, Daniel, "Qualitative Response Models," in Z. Griliches and M. D. Intriligator (eds.), *Handbook of Econometrics*, North Holland, Amsterdam, 1985.

McFadden, Daniel, "Quantal Choice Analysis: A Survey," *Annals of Economic and Social Measurement* 5 (1975), pp. 363-390, 1976.

McFadden, Daniel, "Regression-Based Specification Tests for the Multinomial Logit Model," *Journal of Econometrics* 34 (1987): 63-82.

Mélard, G., "Algorithm AS 197: A Fast Algorithm for the Exact Likelihood of Autoregressive-moving Average Models," *Applied Statistics,* 1984, p.104-109. Code available on Statlib: http://lib.stat.cmu.edu/apstat/

Montalvo, Jose Garcia, "GMM estimation of count-panel-data models with Fixed Effects and Predetermined Instruments," *Journal of Business and Economic Statistics* 15 (1997), pp. 82–89.

Mundlak, Yair, "On the Concept of Non-Significant Functions and Its Implications for Regression Analysis," *Journal of Econometrics* 16(1981), pp. 139-149.

Nawata, Kazumitsu, "Estimation of Sample Selection Models by the Maximum Likelihood Method," *Mathematics and Computers in Simulation* 39 (1995), pp. 299-303.

Nawata, Kazumitsu, "Estimation of Sample Selection Bias Models by the Maximum Likelihood Estimator and Heckman's two-step Estimator," *Economics Letters* 45 (1994), pp. 33-40.

Nawata, Kazumitsu, and Nobuko Nagase, "Estimation of Sample Selection Bias Models," *Econometric Reviews* 15 (1996), pp. 387-400.

Nelson, Charles, Applied Time Series Analysis for Managerial Forecasting, Holden-Day, New York, 1973.

Nelson, C. R., and R. Startz, "Some Further Results on the Exact Small Sample Properties of the Instrumental Variables Estimator," *Econometrica* 58 (1990), pp. 967-976.

Nerlove, Marc, *Likelihood Inference in Econometrics*, Academic Press, 2000.

Nerlove, Marc. “Further Evidence on the Estimation of Dynamic Economic Relations from a Time Series of Cross Sections,” *Econometrica* 39 (1971): 359-382.

Nerlove, Marc and S. James Press, “Univariate and Multivariate Loglinear and Logistic Models,” Rand Report No. R-1306-EDA/NIH, 1973.

Newey, Whitney K., and Kenneth D. West, “A Simple, Positive Semi-definite, Heteroskedasticity and Autocorrelation Consistent Covariance Matrix,” *Econometrica* 55(1987), pp. 703-706.

Olsen, R. J., “Distributional Tests for Selectivity Bias and a More Robust Likelihood Estimator,” *International Economic Review* 23 (1982), pp. 223-240.

Ortega, J.M., and W. C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York, 1970, Chapter 7.

Osterwald-Lenum, Michael, “Practitioners' Corner: A Note with Quantiles for the Asymptotic Distribution of the Maximum Likelihood Cointegration Rank Test Statistic”, *Oxford Bulletin of Economics and Statistics*, 1992, p.461-471.

Pan, Jie-Jian, “Distribution of Noncircular Correlation Coefficients,” *Selected Transactions in Mathematical Statistics and Probability*, 1968, pp. 281-291.

Pantula, Sastry G., Graciela Gonzalez-Farias, and Wayne A. Fuller, “A Comparison of Unit-Root Test Criteria,” *Journal of Business and Economic Statistics* (October 1994): 449-459.

Perron, Pierre, “The Great Crash, The Oil Price Shock, and the Unit Root Hypothesis,” *Econometrica*, November 1989, pp.1361-1401.

Phillips, P. C. B., “Time Series Regression with a Unit Root,” *Econometrica* 55 (1987), pp. 277-301.

Phillips, P. C. B., and Pierre Perron, “Testing for a Unit Root in Time Series Regression,” *Biometrika* (1988): 335-346.

Pindyck, Robert S., and Daniel L. Rubinfeld, *Econometric Models and Economic Forecasts*, McGraw-Hill Book Company, New York, 1966.

Prais, S. J. and Winsten, C. B., “Trend Estimators and Serial Correlation,” Cowles Commission Discussion Paper No. 373, Chicago, 1954.

Quandt, Richard E., “Computational Problems and Methods,” in Griliches and Intriligator (eds.), *Handbook of Econometrics*, Volume I, North-Holland Publishing Company, Amsterdam, 1983.

Rao, C. Radhakrishna, *Linear Statistical Inference and its Applications*, John Wiley and Sons, New York, 1965.

Rao, P., and Zvi Griliches, “Small Sample Properties of Several Two-Stage Regression Methods in the Context of Auto-Correlated Errors,” *JASA* 64(1969), pp. 253-272.

Rothenberg, T. J., "Approximating the Distributions of Econometric Estimators and Test Statistics," Ch. 15 in Z. Griliches and M. Intriligator (eds.), *Handbook of Econometrics, Vol. II*, Amsterdam: North Holland, pp. 881-935.

Rousseeuw, P. J., “Least Median of Squares Regression,” *JASA* 79 (1984), pp. 871-880.

Rousseeuw, P. J., “Progress,” http://win-www.uia.ac.be/u/statis/

Rousseeuw, P. J., and Leroy, A. M., *Robust Regression and Outlier Detection*, Wiley, 1987.

Rousseeuw, P. J., and Wagner, J., “Robust Regression with a distributed intercept using Least Median of Squares,” *Computational Statistics and Data Analysis* 17 (1994), pp. 66-68.

Royal Statistical Society, “Inverse Normal Computation,” Algorithm AS 241, *Applied Statistics* 37 (1988).

Royston, Patrick, “Algorithm AS R94,” *Applied Statistics* 44 (1995).

Saaty, T. L., and J. Bram, *Nonlinear Mathematics*, McGraw-Hill Book Co., New York, 1964.

Savin, N.E., and Kenneth J. White, “Testing for Autocorrelation with Missing Observations.” *Econometrica* 46 (1978): 59-67.

Schaffer, Henry E., Algorithm #369, *Collected Algorithms from ACM,* Volume II, ACM, New York, 1980.

Schmidt, P., S.C. Ahn and D. Wyhowski, “Comment” on “On the Estimation of Panel-Data Models” by Keane, M.P., and D.E. Runkle, *Journal of Business and Economic Statistics* 10 (1992): 10-14.

Sclove, S.L., “Least Squares Problems with Random Regression Coefficients,” Technical Report No. 87, IMSSS, Stanford University, 1973.

“Seasonal Analysis of Economic Time Series,” proceedings of the Conference on the Seasonal Analysis of Economic Time Series, September 1976.

Shapiro, S. S., and M. B. Wilk, “An Analysis of Variance Test for Normality (Complete Samples),” *Biometrika* 52 (1965): 591-611.

Shapiro, S. S., M. B. Wilk, and H. J. Chen, “A Comparative Study of Various Tests of Normality,” *Journal of the American Statistical Association* 63 (1968): 1343-1372.

Shiller, Robert, “A Distributed Lag Estimator Derived from Smoothness Priors,” *Econometrica* 41(1973), pp. 775-787.

Silverman B. W., *Density Estimation for Statistics and Data Analysis*, Chapman and Hall, London, 1986.

Sims, Christopher A., “Macroeconomics and Reality,” *Econometrica* 48(1980), pp. 1-48.

Staiger, D., and J. H. Stock, "Instrumental Variables Regression with Weak Instruments," *Econometrica* 65 (1997), pp. 557-586.

Startz, Richard, “Computation of Linear Hypothesis Tests for Two-Stage Least Squares,” *Economics Letters* 11, 1983, pp. 129-131.

Statlib, http://lib.stat.cmu.edu/apstat/

Steward, D.V., “On an Approach to Techniques for the Analysis of the Structure of Large Systems of Equations,” *SIAM REVIEW*, Volume 4, pp. 321- 342.

Stewart, G. E., Algorithm #384, *Collected Algorithms from ACM* Volume II, ACM, New York, 1980.

Stock, J. H., and M. Yogo, "Testing for Weak Instruments in Linear IV Regression," NBER Technical Working Paper No. 284, October 2002.

Theil, Henri, *Applied Economic Forecasting*, North Holland Publishing Company, 1966.

Theil, Henri, *Economic Forecasts and Policy*, North Holland Publishing Company, 1961.

Theil, Henri, *Principles of Econometrics*, John Wiley & Sons, Inc., New York, 1971.

Thursby, Jerry, *Journal of Econometrics*, 1992.

Tobin, James, “Estimation of Relationships for Limited Dependent Variables,” *Econometrica* 31(1958), pp. 24-36.

Train, Kenneth, *Qualitative Choice Analysis*, The MIT Press, Cambridge, Mass., 1986.

Tsay and Tiao, *JASA*, March 1984, pp. 84-96.

TSP International website: http://www.tspintl.com

Van Daele, Walter, *Applied Time Series and Box-Jenkins Models*, Academic Press, New York, 1983.

Verbeek, Marno, *A Guide to Modern Econometrics*, John Wiley & Sons, Inc., New York, 2000, pp. 189-193.

Wampler, Roy H., “Test Procedures and Test Problems for Least Squares Algorithms,” *Journal of Econometrics* 12, pp 3-21.

White, Halbert, “A Heteroskedasticity-Consistent Covariance Matrix and a Direct Test for Heteroskedasticity”, *Econometrica* 48(1980), pp. 721-746.

White, Halbert, “Instrumental Variables Regression with Independent Observations,” *Econometrica* 50(1982), pp. 483-500.

White, Halbert, “Maximum Likelihood Estimation of Misspecified Models,” *Econometrica* 50(1982), pp. 1-25.

White, Halbert, “Using Least Squares to Approximate Unknown Regression Functions,” *International Economic Review* 21(1980).

Wooldridge, J. M., *Econometric Analysis of Cross Section and Panel Data*, Cambridge, MA: MIT Press, 2002.

Zellner, Arnold, “An Efficient Method of Estimating Seemingly Unrelated Regressions and Tests of Aggregation Bias,” *JASA* 57(1962), pp. 348-368.

Zellner, Arnold, “Estimators for Seemingly Unrelated Regression Equations: Some Exact Finite Sample Result,” *JASA* 58(1963), pp. 977-992.

Zellner, Arnold, and Henri Theil, “Three-Stage Least Squares: Simultaneous Estimation of Simultaneous Equations,” *Econometrica* 30(1962), pp. 54-78.

INDEX