A Quick Start Introduction to NLOGIT 5 and LIMDEP 10
© 1986 - 2012 Econometric Software, Inc. All rights reserved.

This software product, including both the program code and the accompanying documentation, is copyrighted by, and all rights are reserved by Econometric Software, Inc. No part of this product, either the software or the documentation, may be reproduced, stored in a retrieval system, or transmitted in any form or by any means without prior written permission of Econometric Software, Inc.

LIMDEP® and NLOGIT® are registered trademarks of Econometric Software, Inc. All other brand and product names are trademarks or registered trademarks of their respective companies.

Econometric Software, Inc.
15 Gloria Place
Plainview, NY 11803
USA
Tel: +1 516-938-5254
Fax: +1 516-938-2441
Email: sales@limdep.com
Websites: www.limdep.com and www.nlogit.com

Econometric Software, Australia
215 Excelsior Avenue
Castle Hill, NSW 2154
Australia
Tel: +61 (0)4-1843-3057
Fax: +61 (0)2-9899-6674
Email: hgroup@optusnet.com.au
Contents

I. Introduction 5
II. The Desktop: Startup NLOGIT or LIMDEP 6
III. Operating NLOGIT and LIMDEP 7
   A. Data Files 7
   B. Operating with the Menus and Dialogs 7
   C. Using Commands and the Command Editor 12
IV. Stopping, Restarting and Data Sets 15
V. NLOGIT Commands 17
   A. Commands in the Command Editor 17
   B. Names 17
   C. Command Structure 17
VI. Some Essential Operations 19
   A. The Active Sample 19
   B. Missing Values 20
   C. Transformations 20
   D. Variable Lists in Model Commands 21
      1. Categorical Variables 21
      2. Interaction Terms 21
   E. Panel Data 23
   F. Robust covariance Matrices and Cluster Corrections 24
VII Econometric Models 26
   A. Essential Models: Estimation Commands 26
      1. Descriptive Statistics 26
      2. Scatter Plot 27
      3. Histogram 27
      4. Kernel Density Estimator 28
      5. Linear Regression 28
      6. Instrumental Variables – 2SLS 29
      7. Binary Choice 29
      8. Count Data 32
      9. Ordered Choice Models 33
      10. Stochastic Frontier and Data Envelopment Analysis 34
   B. Post Estimation Model Results 36
      1. Predictions 36
      2. Simulations 36
      3. Partial Effects 37
      4. Retained Results 40
   C. Panel Data Forms 41
      1. Fixed Effects Models 41
      2. Random Effects Models 43
      3. Random Parameters Models 43
      4. Latent Class Models 44
VIII. Multinomial Logit and Multinomial Choice  46
A. Data  46
B. Basic Multinomial Choice Models and Choice Substitution Elasticities  47
C. Multinomial Choice Models  50
   1. Multinomial Probit Model  50
   2. Nested Logit Model  51
   3. Mixed (Random Parameters, RP) Logit Model and Willingness to Pay (WTP)  52
D. Stated Choice (Panel) Data  53
   1. Random Parameters Model  54
   2. Error Components (Random Effects) Logit Model  55
   3. Latent Class Multinomial Logit Model  56
IX. Tools  57
A. Scientific Calculator – The CALC Command  57
B. Matrix Algebra  58
C. Procedures  60
D. Bootstrapping  62
E. Displaying Results  64
E. WALD, SIMULATE and Standard Errors for Nonlinear Functions  65
   1. The WALD Command  65
   2. The SIMULATE Command  66
   3. WALD or SIMULATE - Which Should You Use?  66
I. Introduction

This short getting started guide will show you how to operate NLOGIT and LIMDEP. The manuals for NLOGIT and LIMDEP are several thousand pages long, and document hundreds of models, estimators, and other program procedures. This guide will show you how to operate the program and use it to do some of the most common calculations. The program’s interface uses the same basic forms for most of the functions it performs. Based on what we do here, you will be able to construct command streams to do complex analyses using many of the features of the program.

The two programs operate exactly the same way, with the same command set and user interface. NLOGIT 5 is in fact, LIMDEP 10 plus one (extremely large) command set. This short manual will show how to operate both programs. For convenience, the discussion will assume you are using NLOGIT, but everything noted applies equally to LIMDEP as well. A short discussion in Section VIII will introduce the specific difference between NLOGIT and LIMDEP.
II. The Desktop: Startup NLOGIT or LIMDEP

Your program is installed on your computer and you are ready to begin. There is an icon for NLOGIT 5 or LIMDEP 10 on your desktop, and the program is included in your startup menu. Launch your program.

When you first start the program your desktop will look as in Figure 1. (LIMDEP and NLOGIT use the same desktop and functionality. You can see which program you are using by the name that appears at the upper left corner of the desktop. Notice for our discussion here, we are using NLOGIT 5. Operation of the two programs is identical. (The only difference between them is the (large) set of multinomial choice models that are supported in NLOGIT and not in LIMDEP.) We note two small differences that may appear between our desktop in Figure 1 and yours. First, the setting ‘U:38888 Rows: 38888’ appears at the top of the window at the left of our desktop. This is a setting that we (you) can make that relates to how large a data set you want your program to be able to store. A different value will appear the first time you launch the program. Second, the small editing window we call the ‘command bar’ that we have indicated with a red arrow in Figure 1 may not be present on your desktop. You can install this as follows: click Tools→Options→View – note the Tools menu item is above the tip of the red arrow – then click in the check box next to ‘Display Command Bar’ and finally, click OK. This setting is fixed until you change it. Finally, your row of buttons may be above your command bar, not below it. You can move this around the screen as you like.)

The window that is open at the left of the desktop is called the ‘Project Window.’ There is a large amount of functionality operated from this window, as will be clear shortly.

Figure 1. Initial Desktop

A Tip: NLOGIT uses a standard statistical package style, three window mode of operation. The first window you will see is the ‘Project’ window. A project consists of the data you are analyzing and the results of your computations, such as estimates of coefficients, other matrices you might have computed, and so on. As we’ll see shortly, this window contains an inventory of the things you have computed – the inventory will grow as you manipulate your data. You should never close the project window. Nearly all of the program functions operate only when a project is active. You know that a project is active when the project window is present and open. (You can minimize it with the left sizing button, . But, do not close it with the red × button, .)
III. Operating NLOGIT and LIMDEP

NLOGIT provides both menu/dialog boxes and a command language that you can use to operate the program. All of the basic functions of the program can be operated with either. However, many of the more complex operations, including most of the involved models, are accessed only through the command language. We will take a quick look at both of these now.

A. Data Files

You will use NLOGIT to analyze data. To get started, we’ll note a couple things about data. The data you use will have to come from somewhere – probably a public data source, or in a file that you obtained from some external source. (You can create data within NLOGIT, for example, by using the random number generators, but you will rarely do this exclusively. Usually, created data are added to existing data sets.) Data files come in many forms. NLOGIT can read many different kinds of files, and with modern interchange programs such as Stat Transfer, you can convert files from many more sources that might be foreign to NLOGIT to a form that NLOGIT is comfortable with. These issues are discussed in the manual. The most common generic file type used by contemporary researchers is the ‘CSV’ format. A CSV file (i.e., ‘comma separated values’ format) has a line of variable names at the top and rows of data below them, with values separated by commas, such as the data set in Figure 2 below. Figure 2 show the data in a small demonstration file that we will use named IncomeData.csv. (In Figure 2, we are viewing the contents of IncomeData.csv in NLOGIT’s text editor, which we’ll discuss below.)

Figure 2. A CSV File

B. Operating with the Menus and Dialogs

We’ll start by importing the data in IncomeData.csv into the program so that we can analyze them. Select Project→Import→Variables… as shown in Figure 3. This will open a Windows Explorer as shown in Figure 4 that you can use to navigate to your file. Make your way to where the file is installed on your computer. On your computer, this should be in the C:\NLOGIT5 or C:\LIMDEP10 folder. It may be in some other folder depending on how you installed this tutorial on your computer. Select IncomeData.csv in the menu.
After you click Open, the data file will be imported into NLOGIT’s work area and will be ready for you to analyze them. Note in the project window in Figure 1, within the window, in the ‘Data’ area, the first item (folder) is ‘Variables.’ There is nothing at the left of the title, however. After you import your data, the Variables folder will indicate that it contains data, as shown in Figure 5. Note the + next to the folder name.

The project window will now indicate that there are 14 Rows of data – that is the number of observations in the data file that we just read. If you click the + box at the left of Variables to open the folder, the list of variables that have been read will be displayed, as shown in Figure 6. This is our active data set. You can visit the actual data by activating the data editor. The button that will open the data editor is indicated by the red arrow in Figure 6. The spreadsheet style data editor is shown in Figure 7. You can enter and replace data in the editor. After you examine the data editor, you can minimize it or close it. (Closing the data editor only hides the display – it does nothing to the active data set.)
Since the data are ready to use, we will do some computations. From the desktop, select Model→Linear Models→Regression… as shown in Figure 8. This will open a dialog box (Figure 9) that we call a ‘Command Builder.’ You’ll see why momentarily. We’ll build a regression command. First select the dependent variable (INCOME) from the drop down menu as shown in Figure 9. The independent variables are chosen in the windows below the dependent variable. Independent variables are selected by ‘selecting’ them in the right window, then using ‘<<’ to move them to the left window, as shown in Figure 10. Select ONE, AGE and EDUC for our model.

A Tip: ONE is the constant term in the model. NLOGIT does not automatically place a constant term in any model. It must be requested by including ONE (a program created variable) in the list of independent variables.
Figure 8. Model Menu

Figure 9. Command Builder: Dependent Variable
After your model is specified in the command builder – note that there are other options on this page, and two more tabs that promise still more options – press the ‘Run’ button at the lower right of the dialog box. Run asks the program to compute the specified regression. A new window, the ‘Output Window’ opens and displays your regression results. Note above the regression results, there is a line of green text. This is the REGRESS command that was built by the command builder. This is the second window (the data/project window is the first) noted as the three window format. The editing window discussed next is the third.
You have now launched the program, read a data set and computed a regression (without touching your keyboard). Now, we will do the same operations using the NLOGIT’s command language. We’ll start a new session to demonstrate this procedure. Like most other programs, you leave NLOGIT by using File→Exit. The File menu is where it always is in Windows programs, at the upper left of the desktop, and Exit is, as usual, at the bottom of the menu. On your way out, you will be asked about saving the project, Untitled Project 1, and the output window, Untitled Output 1. Click ‘NO’ both times and the program will close.

C. Using Commands and the Command Editor

We will now use NLOGIT’s command language to import the data and compute the regression. Commands are issued by typing them in a text editing window and ‘submitting’ them to NLOGIT’s command processor.

Restart the program as before to produce the empty desktop as in Figure 1. Click File to open the menu as shown in Figure 12. Select ‘New’ at the top of the menu, and the small dialog will open and offer to open a Text/Command Document window or a Project. Select the Text/Command Document option. (You already have a project open.) When you select the Text/Command Document option, the editing window shown in Figure 13 will open.

Figure 12. File Menu and New Dialog

This is the text editor. You can edit anything in it. (You can also have multiple text editing windows open at the same time.) We will enter our commands in this window, then submit them to the command processor.

We want to do two operations right now, import our data file and compute a linear regression. We type the commands in the editor. The two commands that we wish to carry out are ‘IMPORT’ and ‘REGRESS.’ We’ll say more about the commands in a moment. First, type the string ‘import;file=’ in the first line of the window. Now, it’s not sure exactly where the file is on your computer. If you know, you type the path to it after the equals sign. File names are enclosed in double quotes. A $ character is used to end the instruction. (Always, all instructions.) If you don’t know the path to the file, find it as follows: In the desktop menu, select Insert→File Path… and use the Windows explorer to find your file (IncomeData.csv). This will place the file path in the line where you want it, and you need only add the ending $ to complete the command. Press the Enter key. On the next line, type the REGRESS command as shown. Note that it has some parts separated by semicolons and, as always, ends with a $.
A Tip: If you are using a desktop computer with a separate keyboard, use the *alphabetic* Enter key here. The Enter key in the numeric keypad at the right of the keyboard is not the same when you are using *NLOGIT*. We’ll note why shortly.

Now that your two commands are in the editing window, you can submit them. Highlight the two lines of text as if you were about to copy them in an editor such as *Microsoft Word*. When the two lines are highlighted, press the ‘GO’ button that is noted by the red arrow in Figure 14. The output window will appear and will indicate that your file was imported, and the regression was computed.

A Tip: Pressing the numeric keypad’s Enter key is the same as highlighting the one line that the cursor is in and then clicking GO. This is how the two enter keys differ. You can always (and only) submit one line this way.
Figure 15. Commands Executed from Text Editor.

You are not limited to one or the other of these two modes of entering instructions. You can use either the command editor or the menus and dialog boxes whenever you wish. In Figure 16, we have imported the data set using the command editor, then run the regression with the command builder.

Figure 16. Using Both Commands and Dialog Boxes.

A Tip: Notice in Figure 11, the command builder has placed a copy of the command it created in the output window. You can ‘copy’ this command in the output window, ‘paste’ it into the editing window, and submit it again. This would be useful if you want to modify the command, for example by adding more independent variables. (The command processor will ignore the leading ‘|->’ if you happen to include it in your copy of the command.)
IV. Stopping, Restarting and Data Sets

You should only import a data set once. When you exit the program, you are offered a chance to save your data (as any modern program does). For NLOGIT, this is the project. The dialog in Figure 17 will appear when you select File→Exit. The project contains your active data set as well as a long list of other things you create as you operate the program. You can save the project with any name (and at any time) with File→Save Project As... It will be saved as an LPJ file – Windows recognizes this file extension. In Figure 18, we are saving the data in a work folder as IncomeData.lpj. When the program is restarted, instead of importing the original data, we merely load the project. The most recent 4 saved projects will appear in the File menu. In Figure 19, IncomeData.lpj appears in the File menu, and can be selected to resume the analysis of the data. Note in the lower panel of Figure 19, the project file name at the top of the desktop and at the top of the project window is IncomeData.lpj, rather than Untitled Project 1, as it was before.

A Tip: The project will always be current. When you add variables, create new ones, for example by transforming the raw data, the new variables are always saved in the project.

Another Tip: You can launch a project file from Windows Explorer – the same way that selecting a .docx file launches Microsoft Word then imports the document.

A Third Tip: You can also save the text editor as a LIM file. The pair of files constitutes your entire working session. You can resume a session exactly where you were when you exited by reloading these two files.

Figure 17. Saving the Project Upon Exit

Figure 18. Windows Explorer Saving a Project File
Figure 19. Reloading a Project from the File Menu
V. NLOGIT Commands

The menus and dialog boxes are helpful for operating the program. But, they are a bit inconvenient compared to the command editor. Users generally quickly migrate to the command structure for most operations. With that in mind, we will show the basic form of NLOGIT commands, and note some specific ones that you are certain to use. Altogether, there are several hundred different commands and functions. You can operate a large fraction of the program functionality with a few of the most important ones.

A. Commands in the Command Editor

There is a specific protocol for using the text editor to submit commands and a specific format for the commands, themselves. In general, there are very few structures or restrictions. The command language is designed to be convenient and self documenting. Instructions look like what they are requesting. For using the text editor:

- Case almost never matters. Notice in Figure 14, the verb in the REGRESS command is all in capital letters whereas the rest of the command is a mix of caps and lower case. Case only matters for file names and in the titles used for graphics and output tables that you construct (where you would want to use both cases). Otherwise throughout the program, commands can any mix of lower case and upper case letters.

- Spacing only matters in titles and file names. Notice there are some spaces put in the REGRESS command, for clarity. The spaces have no other meaning. In particular, lists of items are always delimited by punctuation, usually commas, never by spaces. You can use spaces in commands anywhere you wish to make them easier to read.

A Tip: You can copy commands out of documents such as Word files and paste them directly into the editor. Tab characters will be treated like spaces. A warning, however, the Word dash character, –, is not the same as an ASCII minus sign. You will generally have to change this manually.

- The number of lines used for a command is arbitrary. Line breaks are used for clarity and ease of interpretation of commands. No special connector is needed to connect the lines of multiple line commands. Some commands for complicated models have many parts, and breaking commands into multiple lines is helpful for self documentation. For example,

  REGRESS ; LHS = income ; RHS = one,age,educ $

  is exactly the same as

  REGRESS
  ; Lhs = income
  ; Rhs = one,age,educ
  $

B. Names

You will create many items, including variables, that have names. Names are limited to 8 characters. The first must be a letter. Allowable characters are letters, digits and the underscore character. Since the program is not case sensitive, different cases of letters do not create different variable names. Of course, since spaces have no meaning, they may not appear in names (they are ignored) There are many types of names used in NLOGIT, including variables, matrices, scalars, synonyms for lists of names, label lists, names used for model definitions, names for output tables, and others. All obey the same conventions.

C. Command Structure

All commands are of the form

  VERB ; information ; information ; … $
Note the two commands in the text editor in Figure 14, IMPORT and REGRESS. There are altogether about 200 verbs that manage files, manipulate the data, fit models and do ancillary computations such as test hypotheses. The common structure is as follows:

- Every command must begin on a new line
- Every command must end with a $ at the end of the last line.
- There is no restriction on how many lines may be used for a command
- There is no restriction on what may be included on specific lines.
- Commands may not have more than 10,000 nonblank characters. You will never come close to this limit.

You may have blank lines in your text editor even in the middle of the commands. Since you submit only the lines you want executed, you may put any other text anywhere you wish in the editor. Explicit comment lines may be inserted by beginning the text with a question mark. E.g.,

```
? This command computes a regression.
REGRESS ; Lhs = income ; Rhs = one,age,educ $
```

A block of lines of text may be marked as comment. For example,

```
/*
   The following commands carry out two regressions.
   The first uses x1.  
   The second uses x1 and x2.
*/
REGRESS ; Lhs = y ; Rhs = one,x1 $
REGRESS ; Lhs = y ; Rhs = one,x1,x2 $
```

This construction would seem to be of marginal usefulness. One way it would be helpful would be for having documentation in command files that you can execute directly with the Run menu Shown in Figure 20.

![Figure 20. Run Menu for Run File...](image)
VI. Some Essential Operations

The following lists a handful of operations that will be part of most analyses.

A. The Active Sample

When you import a data set, the active sample is all the observations in the data set. Figure 21 shows the income data we are examining in our demonstration. There are 14 observations in the data set. Note, the 14 rows are numbered and there is a chevron (») in each row. The » indicates that the observation is in the 'current sample.' The active sample can be changed in several ways. Three commands, SAMPLE, REJECT, and INCLUDE are used specifically to change the sample.

SAMPLE ; n1 – n2 $

sets the sample to be rows n1 to row n2. For example,

SAMPLE ; 4 – 12 $

in the example would select the observations shown in Figure 21b. Note the chevrons are now only present for the active subset of the data. The excluded observations are not lost. But, any operation that manipulates the data set operates only on these observations. The full sample is restored with

SAMPLE ; All $

The two other commands used directly to change the sample are

REJECT ; condition $ such as REJECT ; age > 60 $,

which removes observations from the active sample, whatever it happens to be, and

INCLUDE ; condition $ such as INCLUDE ; female = 1 $

which adds observations to the current sample, whatever it happens to be. These commands can be applied to the full data set, or no data set, respectively, by including ;New. For example,
INCLUDE ; New ; Female = 1 $

starts with no observations, then adds to the empty data set all observations in the full data set that have Female = 1.

A Tip: In many cases, you will want to fit a model using a subset of the active data set, but not wish actually to change the active data set. A model command can do that automatically. For example,

\[ \text{REGRESS} \quad \text{if} \ [ \text{age} < 60 ] \quad \text{Lhs} = \text{income} \quad \text{Rhs} = \text{one,age,educ} \]

B. Missing Values

The internal missing value code is -999. In the data editor, -999 will appear as a blank. In general, you must inform \textit{NLOGIT} what to do about missing values. In general, \textit{NLOGIT} only acts on missing data when you ask it to do so. If your sample contains missing values and you make no indication, the -999s will be treated as ordinary data. A global command to tell the program to bypass missing values when it fits models is

\[ \text{SKIP} \]

In the desktop, you can use Project\textrightarrow{Settings}--Execution and check the box for skipping missing data. \textit{SKIP} is a fixed setting. It persists from model to model. You can turn it off with \textit{NOSKIP} if you wish.

A Tip: \textit{NLOGIT} contains a large package for multiple imputation of missing values.

C. Transformations

You will usually want to compute transformed variables. The command is

\[ \text{CREATE} \quad \text{variable} = \text{expression} \quad \text{variable} = \text{expression} \quad \ldots \]

The left hand variable may be a new variable created from existing variable(s) or may be an existing variable, which will be replaced. For example,

\[ \text{CREATE} \quad \text{logincm} = \log(\text{income}) \quad \text{agesq} = \text{age}^2 / 100 \]

A common calculation is creating dummy variables. There are many ways to do so. For example, two ways to create the variable YOUNG equal to 1 if AGE is less than 25 and 0 otherwise would be

\[ \text{CREATE} \quad \text{young} = \text{age} < 50 \]

and

\[ \text{CREATE} \quad \text{if(age < 50)young} = 1 \]

Note that the log income variable uses a function, \textit{log}(.). There are over 200 functions supported, including \textit{log, exp, abs, min,} and many special functions. All functions have 3 character names. \textit{NLOGIT} contains 20 different random number generators, such as Rnn(mean,standard deviation) which computes a random sample of observations from a normal distribution with the indicated mean and standard deviation. Functions may appear in expressions. For example, to create a sample of observations from the \textit{F} distribution with 5 and 27 numerator and denominator degrees of freedom, you might use

\[ \text{CREATE} \quad \text{fsample} = (\text{rnx(5)/5}) / (\text{rnx(27)/27}) \]

(But, this would be the same as \textit{Rnf(5,27)}.)

The seed for the generators is set using

\[ \text{CALC} \quad \text{ran(value)} \]

You will use this to be able to replicate your analyses that use random values.
D. Variable Lists in Model Commands

Model commands contain lists of variables. The lists can be extremely long – possibly hundreds of variables. There are several shortcuts provided. The primary device is

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

For example,

```
NAMELIST ; x = one,age,educ,female $
REGRESS ; Lhs = income ; Rhs = x $
```

Namelist provide a convenient shortcut for model commands. They also serve many other functions. One major one is defining data matrices. For example, to compute ‘by hand’ the least squares coefficient vector that is reported by `REGRESS` above, we could use

```
MATRIX ; bols = <x’x> * x’income $
```

The construction `<matrix>` is `NLOGIT`'s syntax for computing the inverse of a matrix. Note that the namelist and the variable become a data matrix and a data vector when used in a matrix command.

1. Categorical Variables

Categorical variables are often used in models in the form of a set of dummy variables with one of the dummy variables being dropped as the ‘base case.’ In the example below, rather than use `EDUC` in years, we have used `RECODE` to create a category variable `ED` which is 0 when `EDUC` is 0-9, 1 when `EDUC` is 10-12, and 2 when `EDUC` is 13-20. The regression would then use dummy variables for the second and third categories. A special format, `#name`, is used for category variables. It is not necessary actually to compute the dummy variables. Note the results in Figure 2, which reports how the variable `#ED` has been used in the regression.

![Figure 2. Categorical Variable in Regression](image)

2. Interaction Terms

A second common feature of models is ‘interaction terms.’ In the model results in Figure 23, we have included education, female, and an interaction between education and female. Note that the command contains the interaction. We do this rather than computing a product variable, say `CREATE;EducFemale=Educ*Female$`
A Tip: Namelists may contain interactions. For example,

```
NAMELIST ; EdFem= female,educ*female $
REGRESS ; Lhs = income ; Rhs = one,age,edfem $
```

Note that EdFem is not a variable. It is a list of three variables, one of which is a product of two variables. You can also include the interaction terms directly in the model command, as shown in Figure 23.

![Figure 23. Interaction Effect in a Model](image)

It is possible as well to have interactions of categorical variables and other variables, as shown in Figure 24. Although it is unlikely that you would need it, it is also possible to have interactions of categorical variables. The procedure is described in the manual.

![Figure 24. Interaction of Categorical Variable with Other Variable](image)

A Tip: It is easy to create multicollinearity with category variables and interactions. In all cases, NLOGIT will do its best to compute the regression you specify. NLOGIT will never, upon detecting multicollinearity, drop some variables and fit some model that you did not specify that does not have a multicollinearity problem. Decisions about model specification are made by you, not the program.
E. Panel Data

All panel data applications are handled the same way. To set up the procedures, you will prepare an indicator variable that \textit{NLOGIT} will use to manage the data handling. Our 14 observation, IncomeData file is a panel, as can be seen from the ID variable in Figure 25. \textit{NLOGIT} assumes that a panel data set contains some kind of identifier variable such as ID in Figure 25. The ID variable does not have to be a sequential set of integers. It can be anything (it need not even be integers), so long as it takes the same value for every observation in a group and it changes (up or down) from one group to the next. To set up a panel, at the beginning of your session, use

\begin{verbatim}
SETPANEL ; Group = the id variable ; Pds = name for a variable that \textit{NLOGIT} will now create $
\end{verbatim}

The Pds variable will contain in each row of a group the number of observations in the group. You may use any name you wish. We usually use Ti. Figure 25 shows the results of

\begin{verbatim}
SETPANEL ; Group = id ; Pds = Ti $
\end{verbatim}

\textbf{NOTE:} \textit{SETPANEL} should be issued immediately after the data are imported.

A Tip: This works the same way for balanced or unbalanced panels. You need not worry about unbalanced panels.

A Second Tip: If you have a balanced panel with \textit{T} periods (whatever \textit{T} is) and you don’t have an ID variable, you can create one with

\begin{verbatim}
CREATE ; MyID = Trn(T,0) $
\end{verbatim}

A third Tip: If you have an unbalanced panel and you do not have an ID variable, you cannot use this data set as a panel. You must create the ID variable somehow. \textit{NLOGIT} cannot do it for you.

A Last Tip: The \textit{SETPANEL} setting is not etched into the project. When you save the project, \textit{SETPANEL} is not saved. When you reload the project, you must reissue the \textit{SETPANEL} command.

\textit{SETPANEL} creates some internal settings as well. Most panel versions of models are requested by just adding \textit{;PANEL} to the model command. If you change the sample from what it was when you issued the \textit{SETPANEL} command, this will break the counter variable. Not to worry. When your command contains \textit{;PANEL}, \textit{NLOGIT} recreates the counter so that it matches the observations in the active sample. In our example, if we were to \texttt{REJECT;Age>62$}, the count for group 3 would be incorrect – it would change from 4 to 3. \textit{SETPANEL} takes care of this as it processes your model commands.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{PanelData.png}
\caption{Panel Data}
\end{figure}
After the panel data set is defined with SETPANEL, the panel data versions of most models are invoked just by adding ;Panel to the command, as shown in Figure 26.

![Figure 26. Panel Data Regression Command](image)

The default linear panel data model produces quite a lot of results – it displays all three of the pooled model, fixed effects and random effects estimates. The Figures 27 and 28 show the preliminary results and the fixed effects results for our small data set. You can specialize the command with

```latex
REGRESS ; Lhs = income ; Rhs = one,age,female ; Panel ; Fixed Effects $
```

and likewise for random effects. All of the panel data models in NLOGIT (there are about 50) provide several versions (e.g., fixed vs. random effects) that are requested by adding ;Panel and an additional specification in the model command.

```
|-> SETPANEL ; Group = id ; Pds = ti $
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable =</td>
<td>Group sizes</td>
<td>Max</td>
<td>Min</td>
<td>Average</td>
</tr>
<tr>
<td>TI</td>
<td>ID</td>
<td>5</td>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>
|-> REGRESS ; Lhs = income ; Rhs = one,age,female;panel $
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable =</td>
<td>Group sizes</td>
<td>Max</td>
<td>Min</td>
<td>Average</td>
</tr>
<tr>
<td>TI</td>
<td>ID</td>
<td>5</td>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>
```

Figure 27. Preliminary Report for Panel Data Model

**F. Robust Covariance Matrices and Cluster Corrections**

We mention this feature separately because it is so common in the contemporary literature. So called robust covariance matrices for least squares and maximum likelihood estimators are requested by using

```latex
; Robust
```

in the model command.

A Tip: The ‘robust’ for the linear model in a cross section is the White estimator, which is requested (only for the linear model) with ;Heteroscedasticity. For time series, the Newey-West estimator is requested with ;Pds=T without ;Panel.
The correction for clustering is applied in panel data sets (or clustered data sets that look like panels). All model commands are modified the same way:

; Cluster = an identity variable such as ID in figures 29 and 30

or

; Cluster = a fixed cluster size if all clusters are the same size, e.g., ; Cluster = 5.

---

A Tip: ; Cluster is supported for every model that is estimated using least squares or maximum likelihood estimation. It is not supported for quantile estimators or nonparametric estimators.
VII. Econometric Models

There are several hundred model specifications supported by NLOGIT. The set has roughly 70 basic forms such as linear regression, Poisson, Logit, Tobit, and so on. Nearly all of the basic specifications support multiple variants and extensions and about 50 also support several different panel data treatments. For example, Poisson also includes 5 forms of negative binomial models and several additional forms of count models, as well as fixed effects, random effects, latent class and random parameters specifications. Probit is the basic binary choice model, but you can also choose among 5 other forms including Logit, Arctangent, Weibull, Complementary log log, and some exotic forms that few people have ever heard of but are useful for studying the behavior of binary choice estimators. The list below will show the commands for some of the most common and familiar models. In each case, there are many variants described in the manual. And, of course, there are the hundreds of additional models. All of the models listed below are contained in both LIMDEP and NLOGIT. In each case there are many options that can be added to the model command. The list below shows a few in each case.

In the discussions to follow, we will present some examples based on a larger, ‘real’ data set named HealthData.csv. This is a subset of a larger data set from a health economics study by Riphahn, Wambach and Million that appeared in the Journal of Applied Econometrics in 2003. The original panel data set contains 27,326 observations on 7,293 households. Our subset contains 2,039 observations on 550 households. The data are imported with the usual command

```
IMPORT; File="… HealthData.csv"
```

The discussions below show the results of various commands that illustrate the models. There is a script file for you to use to enter the commands by highlighting them one at a time. Use File→Open... and navigate to HealthData.lim to open the file in its own Text/Command window.

A. Essential Models: Estimation Commands

These are some of the most commonly used models and data analysis tools:

1. Descriptive statistics

```
DSTAT ; Rhs = the list of variables $ 
Useful options ; Output = 2 requests a correlation matrix 
; Str = categorical variable requests statistics by strata 
; Quantiles requests order statistics for each variable 
; Rhs = * requests results for all variables.
```

Example: DSTAT ; Rhs = *

<table>
<thead>
<tr>
<th>Descriptive Statistics for 15 variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable</td>
</tr>
<tr>
<td>----------</td>
</tr>
<tr>
<td>ID</td>
</tr>
<tr>
<td>AGE</td>
</tr>
<tr>
<td>EDUC</td>
</tr>
<tr>
<td>FEMALE</td>
</tr>
<tr>
<td>MARRIED</td>
</tr>
<tr>
<td>HHSIZE</td>
</tr>
<tr>
<td>INCOME</td>
</tr>
<tr>
<td>DOVCNT</td>
</tr>
<tr>
<td>HOSPITAL</td>
</tr>
<tr>
<td>PUBLIC</td>
</tr>
<tr>
<td>ACTION</td>
</tr>
<tr>
<td>DOCTOR</td>
</tr>
<tr>
<td>HOSPITAL</td>
</tr>
<tr>
<td>HEALTHY</td>
</tr>
<tr>
<td>HEALTHY</td>
</tr>
</tbody>
</table>

DSTAT results are matrix LASTDSTA in current project.

Figure 31. Results for DSTAT
2. Scatter plot

```
PLOT ; Lhs = variable on horizontal axis
; Rhs = variable(s) on vertical axis
Useful options:
; Title=Up to 80 characters for title
; Vaxis=Up to 60 characters for vertical axis
; Grid to request background grid
; Fill to request lines to connect dots in plot
; Regression to display regression line of Rhs variable on Lhs variable
Example: PLOT ; if[(Income <= 1.25]
; Lhs=educ
; Rhs=income
; Title=Income vs. Education (Income Under 1.25)
; Grid ; Regression
```

![Figure 32. Scatter plot with Regression](image)

3. Histogram

```
HISTOGRAM ; Rhs = the variable
Useful options
; Title=up to 80 characters for title
; Group = a categorical variable that defines up to 5 groups
Example: HISTOGRAM ; if[(income <= 1.25] ; Rhs = hlthsat
; Title=Health Satisfaction by Gender
; Group = Female ; Labels=Male,Female
```

![Figure 33. Histogram for Two Groups](image)
4. Kernel Density Estimator

```
KERNEL ; Rhs = list of variable(s) (up to 5) $

Useful options ; Normal – plots normal density with same mean and variance
                  ; Title=up to 80 characters for title

Example: KERNEL ; if[income <= 1.25] ; Rhs = Income
                  ; Title=Income by Gender
                  ; Group = Female ; Labels=Male,Female $
```

Figure 34. Kernel Density Estimators

5. Linear Regression

```
REGRESS ; Lhs = dependent variable
          ; Rhs = independent variables (include constant term ONE on Rhs) $

Useful options ; Cluster = specification
                  ; Heteroscedasticity to request White estimator
                  ; Plot to request a plot of residuals
                  ; Test: restrictions.
                  ; Test: list of variables tests the hypothesis that the coefficients are all zero

Example: REGRESS ; Lhs = income
                  ; Rhs = one,age,educ,married,female,hhkids
                  ; Cluster = id $
```

Figure 35. Linear Regression with Cluster Corrected Standard Errors
6. Instrumental Variables – 2SLS

2SLS ; Lhs = dependent variable ; Rhs = all right hand side variables ; Inst = list of all exogenous variables including all exogenous variables (and ONE) that are in the Rhs list plus any instrumental variables not in the model $

Useful options: ; Cluster

Example: 2SLS ; Lhs = income ; Rhs = one,age,educ,hlthsat ; Inst = one,age,educ,married,hhkids $

Two stage least squares regression ............
LHS = INCOME
Mean = .34950
Standard deviation = .16611
Number of observ. = 2039
Model size Parameters = 4
Degrees of freedom = 2035
Sum of squares = 421.241
Residual = Standard error of e = .45497
Fit R-squared = .80582
Adjusted R-squared = .80589

Not using OLS or no constant. Regd & P may be < 0
Not Instrumental Variables:

<table>
<thead>
<tr>
<th></th>
<th>Coefficient</th>
<th>Standard Error</th>
<th>Prob.</th>
<th>95% Confidence Interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>ONE</td>
<td>1.16576***</td>
<td>.31967</td>
<td>3.65</td>
<td>.0003</td>
</tr>
<tr>
<td>AGE</td>
<td>-.00830***</td>
<td>.00279</td>
<td>-2.97</td>
<td>.0030</td>
</tr>
<tr>
<td>EDUC</td>
<td>-.04170***</td>
<td>.00838</td>
<td>4.97</td>
<td>.0000</td>
</tr>
<tr>
<td>HLTHSAT</td>
<td>-.30735***</td>
<td>.11547</td>
<td>-3.35</td>
<td>.0008</td>
</tr>
</tbody>
</table>

Note: *** , **, * => Significance at 1%, 5%, 10% level.

Figure 36. Two Stage Least Squares

7. Binary Choice

PROBIT or LOGIT ; Lhs = dependent variable ; Rhs = independent variables $

Useful options ; Hold requests results be retained for use by SELECTION in the next step

Example:

SET PANEL ; Group = id ; Pds = Ti $

PROBIT ; Lhs = doctor ; Rhs = one,age,educ,married,female,income ; Panel ; RandomEffects ; Test: income $
Binomial Probit Model
Dependent variable: DOCTOR
Log likelihood function: -1296.76266
Restricted log likelihood: -1310.02091
Chi squared [v.d.f.]: 98.51611
Significance level: .000000
McFadden Pseudo R-squared: .0365953
Estimation based on N = 2039, K = 5
Inf. Cr. AIC = 2608.5 AIC/N = 1.279
Hosmer-Lemeshow chi-squared = 15.57925
P-value: .05490 with deg. fr. = 8
----- LM test for Random Effects -----  
Chi^2adj[1] = 84.927 P-value: .00000

<table>
<thead>
<tr>
<th>Standard</th>
<th>Prob.</th>
<th>95% Confidence Interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constant</td>
<td>.17085</td>
<td>.29921  - .82  .4347</td>
</tr>
<tr>
<td>AGE</td>
<td>.01928**</td>
<td>.00283  .0066</td>
</tr>
<tr>
<td>EDUC</td>
<td>.05447**</td>
<td>.01384  -2.64 .00627</td>
</tr>
<tr>
<td>MARRIED</td>
<td>.12723</td>
<td>.07698  1.09  .0882</td>
</tr>
<tr>
<td>FEMALE</td>
<td>.20246**</td>
<td>.01016  .06000  .02796</td>
</tr>
<tr>
<td>INCOME</td>
<td>-.12679</td>
<td>.16113  - .60 .4956</td>
</tr>
</tbody>
</table>

Note: **, *, ** = Significance at 1%, 5%, 10% level.

Random Effects Binary Probit Model
Dependent variable: DOCTOR
Log likelihood function: -1188.67224
Restricted log likelihood: -1199.76290
Chi squared [v.d.f.]: 232.16724
Significance level: .00000

Inf. Cr. AIC = 2575.4 AIC/N = 1.165
Unbalanced panel has 550 individuals
- Chi^2adj[1] tests for random effects
LM Chi^2adj = 84.927 P-value: .00000
LR Chi^2adj = 232.167 P-value: .00000
Wald Chi^2adj = 166.237 P-value: .00000
Wald test of 1 linear restrictions
Chi^2adj = .02, P-value: .778

<table>
<thead>
<tr>
<th>Standard</th>
<th>Prob.</th>
<th>95% Confidence Interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constant</td>
<td>.34398</td>
<td>.38114  - .96 .3680</td>
</tr>
<tr>
<td>AGE</td>
<td>.02426**</td>
<td>.00519  4.68 .00000</td>
</tr>
<tr>
<td>EDUC</td>
<td>-.04679</td>
<td>.02430  -1.92 .0643</td>
</tr>
<tr>
<td>MARRIED</td>
<td>.03880</td>
<td>.11837  3.10  .76057</td>
</tr>
<tr>
<td>FEMALE</td>
<td>.51913**</td>
<td>.11486  4.52 .00000</td>
</tr>
<tr>
<td>INCOME</td>
<td>.04731</td>
<td>.07761  1.15 .0778</td>
</tr>
<tr>
<td>Rho</td>
<td>.48886**</td>
<td>.03751  12.89 .06000</td>
</tr>
</tbody>
</table>

Note: **, *, ** = Significance at 1%, 5%, 10% level.

Figure 37. Random Effects Probit with Test of Restriction
Example LOGIT ; Lhs = doctor
 ; Rhs = one, age, educ, married, female, income
 ; ROC $
8. Count Data

**POISSON**; Lhs = dependent variable

**Useful options**; Exposure = exposure variable

**NEGBIN**; same as Poisson for negative binomial model

**Useful options**; Model = NB1 or NB2 or NBP

**Example:** NEGBIN; Lhs = docvis; Rhs = one, age, educ, married, income

---

**Table 1:**

<table>
<thead>
<tr>
<th>Coefficient (Standard Error)</th>
<th>z</th>
<th>Prob.</th>
<th>95% Confidence Interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constant</td>
<td>1.56905***</td>
<td>.00991</td>
<td>1.770</td>
</tr>
<tr>
<td>AGE</td>
<td>.02110***</td>
<td>.00107</td>
<td>19.74</td>
</tr>
<tr>
<td>EDUCATION</td>
<td>-.06077***</td>
<td>.00331</td>
<td>-18.01</td>
</tr>
<tr>
<td>MARRIED</td>
<td>.05290***</td>
<td>.00279</td>
<td>19.02</td>
</tr>
<tr>
<td>INCOME</td>
<td>-.45474***</td>
<td>.00400</td>
<td>-113.5</td>
</tr>
</tbody>
</table>

**Note:** ***, **, * --> Significance at 1%, 5%, 10% level.

---

**Table 2:**

<table>
<thead>
<tr>
<th>Coefficient (Standard Error)</th>
<th>z</th>
<th>Prob.</th>
<th>95% Confidence Interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constant</td>
<td>1.71407***</td>
<td>.02436</td>
<td>2.04</td>
</tr>
<tr>
<td>AGE</td>
<td>.02081***</td>
<td>.00320</td>
<td>6.50</td>
</tr>
<tr>
<td>EDUCATION</td>
<td>-.06684***</td>
<td>.00442</td>
<td>-14.72</td>
</tr>
<tr>
<td>MARRIED</td>
<td>.21360***</td>
<td>.00749</td>
<td>2.86</td>
</tr>
<tr>
<td>INCOME</td>
<td>-.68861***</td>
<td>.00915</td>
<td>-75.95</td>
</tr>
</tbody>
</table>

**Note:** ***, **, * --> Significance at 1%, 5%, 10% level.

---

Figure 39. Negative Binomial Model with Poisson Starting Values
9. Ordered Choice Models

ORDERED or OPROBIT ; Lhs = dependent variable
Useful options ; LOGIT for ordered logit. Verb OLOGIT is the same as ORDERED;Logit.
Example: ORDERED ; Lhs = hlthsat
           ; Rhs = one,age,educ,income,female
           ; Partial ; Full $

Ordered Probability Model
Dependent variable   HLTHSAT
Log likelihood function -2861.66224
Restricted log likelihood -2934.00920
Chi squared [ 4 d.f.] 146.29382
Significance level .00000
Multinomial Pseudo r-squared .2249239
Estimation based on N = 2039, K = 0
Inf.Cr.AIC = 579.3 AIC/N = 2.015
Model estimated: Apr 21, 2013, 00:36:39
Underlying probabilities based on Normal

<table>
<thead>
<tr>
<th>HUTHSAT</th>
<th>Coefficient</th>
<th>Standard Error</th>
<th>t</th>
<th>Prob.</th>
<th>95% Confidence Interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constant</td>
<td>1.50559**</td>
<td>0.1722</td>
<td>8.80</td>
<td>.0000</td>
<td>1.06892 - 2.94218</td>
</tr>
<tr>
<td>AGE</td>
<td>-0.02280**</td>
<td>0.0072</td>
<td>-3.21</td>
<td>.0014</td>
<td>-0.0367 - -0.0089</td>
</tr>
<tr>
<td>EDUC</td>
<td>0.05454***</td>
<td>0.0186</td>
<td>2.93</td>
<td>.0037</td>
<td>0.01903 - 0.0898</td>
</tr>
<tr>
<td>INCOME</td>
<td>0.21960</td>
<td>0.1449</td>
<td>1.50</td>
<td>.1336</td>
<td>-0.06751 - 0.50671</td>
</tr>
<tr>
<td>FEMALE</td>
<td>-0.04029</td>
<td>0.0479</td>
<td>-0.84</td>
<td>.4002</td>
<td>-0.13415 - 0.05357</td>
</tr>
</tbody>
</table>

Note: **, *** Significant at 1%, 5%, 10% level.

| CELL FREQUENCIES FOR ORDERED CHOICES |
|---------------------------|-----------------|-----------------|-----------------|-----------------|
| Outcome | Frequency | Cumulative < = | Cumulative > = |
|---------------------------|-----------------|-----------------|-----------------|-----------------|
| HUTHSAT=01 | 125 | 6.1305 | 125 | 6.1305 |
| HUTHSAT=02 | 217 | 10.6425 | 342 | 16.7729 |
| HUTHSAT=03 | 812 | 38.6234 | 1154 | 56.5964 |

| Cross table of predictions and actual outcomes |
|---------------------------|-----------------|-----------------|-----------------|-----------------|
| y(i,j) | 0 | 1 | 2 | 3 | 4 | Total |
|---------------------------|-----------------|-----------------|-----------------|-----------------|
| 0 | 125 | 0 | 1 | 2 | 127 |
| 1 | 205 | 0 | 122 | 217 | 322 |
| 2 | 581 | 0 | 122 | 403 | 704 |
| 3 | 391 | 0 | 59 | 439 | 939 |
| 4 | 0 | 391 | 0 | 59 | 446 |
| Total | 1849 | 0 | 190 | 2039 |

Row = actual, Column = Prediction, Model = Probit
Prediction is number of the most probable cell.

| Cross table of outcomes and predicted probabilities |
|---------------------------|-----------------|-----------------|-----------------|-----------------|
| y(i,j) | 0 | 1 | 2 | 3 | 4 | Total |
|---------------------------|-----------------|-----------------|-----------------|-----------------|
| 0 | 125 | 53 | 23 | 222 | 125 |
| 1 | 205 | 90 | 44 | 249 | 249 |
| 2 | 581 | 171 | 166 | 812 | 812 |
| 3 | 391 | 103 | 103 | 499 | 499 |
| 4 | 0 | 103 | 103 | 206 | 206 |
| Total | 1849 | 422 | 422 | 444 | 2039 |

Row = actual, Column = Prediction, Model = Probit
Value(1) = Sum(i,j) y(i,j)*pi(j).
Column totals may not match cell sums because of rounding error.

Figure 40. Estimated Ordered Probit Model
Figure 41. Full Partial Effects Analysis for Ordered Probit

### Summary of Marginal Effects for Ordered Probability Model (prob)$

Effects computed at means. Effects for binary variables ($\dagger$) are computed differences of probabilities, other variables at means. Binary variables change only by 1 unit so m.d. changes are not shown. Elasticities for binary variables partial effect/probability = "chge".

<table>
<thead>
<tr>
<th>Outcome</th>
<th>Continuous Variable AGE</th>
<th>Changes in AGE</th>
<th>% chg</th>
<th>1 StdDev</th>
<th>Low to High</th>
<th>Elast</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y = 00</td>
<td>0.0245</td>
<td>0.0354</td>
<td>1.38 5</td>
<td>-0.0002</td>
<td>-0.0001</td>
<td>0.096</td>
</tr>
<tr>
<td>Y = 01</td>
<td>0.0293</td>
<td>0.0301</td>
<td>1.04 4</td>
<td>-0.0001</td>
<td>-0.0001</td>
<td>0.045</td>
</tr>
<tr>
<td>Y = 02</td>
<td>0.0272</td>
<td>0.0292</td>
<td>1.02 1</td>
<td>-0.0001</td>
<td>-0.0001</td>
<td>0.043</td>
</tr>
<tr>
<td>Y = 03</td>
<td>0.0233</td>
<td>0.0255</td>
<td>1.01 7</td>
<td>-0.0001</td>
<td>-0.0001</td>
<td>0.041</td>
</tr>
<tr>
<td>Y = 04</td>
<td>0.0232</td>
<td>0.0255</td>
<td>1.01 7</td>
<td>-0.0001</td>
<td>-0.0001</td>
<td>0.041</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Outcome</th>
<th>Continuous Variable EDU</th>
<th>Changes in EDU</th>
<th>% chg</th>
<th>1 StdDev</th>
<th>Low to High</th>
<th>Elast</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y = 00</td>
<td>-0.0131</td>
<td>-0.0131</td>
<td>1.00 0</td>
<td>-0.0212</td>
<td>-0.0212</td>
<td>0.055</td>
</tr>
<tr>
<td>Y = 01</td>
<td>-0.0121</td>
<td>-0.0121</td>
<td>1.00 0</td>
<td>-0.0212</td>
<td>-0.0212</td>
<td>0.055</td>
</tr>
<tr>
<td>Y = 02</td>
<td>-0.0111</td>
<td>-0.0111</td>
<td>1.00 0</td>
<td>-0.0212</td>
<td>-0.0212</td>
<td>0.055</td>
</tr>
<tr>
<td>Y = 03</td>
<td>-0.0101</td>
<td>-0.0101</td>
<td>1.00 0</td>
<td>-0.0212</td>
<td>-0.0212</td>
<td>0.055</td>
</tr>
<tr>
<td>Y = 04</td>
<td>-0.0101</td>
<td>-0.0101</td>
<td>1.00 0</td>
<td>-0.0212</td>
<td>-0.0212</td>
<td>0.055</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Outcome</th>
<th>Continuous Variable INCOME</th>
<th>Changes in INCOME</th>
<th>% chg</th>
<th>1 StdDev</th>
<th>Low to High</th>
<th>Elast</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y = 00</td>
<td>-0.0859</td>
<td>-0.0859</td>
<td>1.00 0</td>
<td>-0.1271</td>
<td>-0.1271</td>
<td>0.168</td>
</tr>
<tr>
<td>Y = 01</td>
<td>-0.0871</td>
<td>-0.0871</td>
<td>1.00 0</td>
<td>-0.1271</td>
<td>-0.1271</td>
<td>0.168</td>
</tr>
<tr>
<td>Y = 02</td>
<td>-0.0882</td>
<td>-0.0882</td>
<td>1.00 0</td>
<td>-0.1271</td>
<td>-0.1271</td>
<td>0.168</td>
</tr>
<tr>
<td>Y = 03</td>
<td>-0.0890</td>
<td>-0.0890</td>
<td>1.00 0</td>
<td>-0.1271</td>
<td>-0.1271</td>
<td>0.168</td>
</tr>
<tr>
<td>Y = 04</td>
<td>-0.0890</td>
<td>-0.0890</td>
<td>1.00 0</td>
<td>-0.1271</td>
<td>-0.1271</td>
<td>0.168</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Outcome</th>
<th>Binary(0-1) Variable FEMALE</th>
<th>Changes in #FEMALE</th>
<th>% chg</th>
<th>1 StdDev</th>
<th>Low to High</th>
<th>Elast</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y = 00</td>
<td>0.0000</td>
<td>0.0000</td>
<td>1.00 0</td>
<td>-0.0001</td>
<td>-0.0001</td>
<td>0.000</td>
</tr>
<tr>
<td>Y = 01</td>
<td>0.0000</td>
<td>0.0000</td>
<td>1.00 0</td>
<td>-0.0001</td>
<td>-0.0001</td>
<td>0.000</td>
</tr>
<tr>
<td>Y = 02</td>
<td>0.0000</td>
<td>0.0000</td>
<td>1.00 0</td>
<td>-0.0001</td>
<td>-0.0001</td>
<td>0.000</td>
</tr>
<tr>
<td>Y = 03</td>
<td>0.0000</td>
<td>0.0000</td>
<td>1.00 0</td>
<td>-0.0001</td>
<td>-0.0001</td>
<td>0.000</td>
</tr>
<tr>
<td>Y = 04</td>
<td>0.0000</td>
<td>0.0000</td>
<td>1.00 0</td>
<td>-0.0001</td>
<td>-0.0001</td>
<td>0.000</td>
</tr>
</tbody>
</table>

### FRONTIER

- **Lhs** = dependent variable
- **Rhs** = independent variables

#### Useful options

- Cost to fit cost frontier. Production is the default
- Techeff = variable to hold estimate of technical efficiency, firm specific
- Eff = variable to hold estimate of inefficiency, firm specific
- ALG = DEA requests data envelopment analysis

#### Example

This example uses a data set on production of Spanish Dairy farms.

- NAMELIST; x = one,x1,x2,x3,x4
- REGRESS; quietly; Lhs = yit; Rhs = x; Res = ols

---

### Statistical Analysis

10. Stochastic Frontier and Data Envelopment Analysis
Figure 42. Stochastic Frontier Efficiency Analysis
B. Post Estimation Model Results

After estimation of the model parameters, a variety of computations are used to analyze the model results. Two common calculations are model predictions/simulations and partial effects. There are also standard results computed with model results that are retained so that they can be used in later analyses.

1. Predictions

Single equation models can create a new variable that is the predictions for the model using

; Keep = the new variable.

Models differ on what the prediction is. In most cases, it is the expected value of the dependent variable. For a few models, it is also possible to retain residuals with

; Res = the new variable.

For most models, this is not a meaningful result, however. For probability models, such as PROBIT, LOGIT and ORDERED, the predicted probability for the observed outcome is saved with

; Prob = the new variable.

2. Simulations

After estimation, model estimation programs store the results for two large processors to use, the simulator and the partial effects program. These use separate post estimation commands. The simple command

SIMULATE $

Produces the average prediction from the model, with an estimated standard error and confidence interval for the mean simulation. Adding ;List to the SIMULATE command produces a listing of the predictions. Adding ;Keep=name to the command requests that a new variable that contains the simulated values be created in the data set.

A Tip: If you have thousands of observations, it might not be a good idea to use ;List. If you are trying to produce what looks like a huge list, the program will ask you if you are sure you want to do this.

Figure 43 shows estimation and simulation of a linear regression with an interaction term. The SIMULATE feature accounts for the nonlinearities in the regression. A second example based on a binary choice model appears below in Figure 44.
3. Partial Effects

Partial effects are an essential part of model estimation. There are several issues to be considered in computing partial effects for a nonlinear model:

- **Partial effects are often (correctly) computed as scaled coefficients.** However, differences can arise between results computed by using the sample means of the data (PEA, or partial effects at averages) and results computed by averaging the computations across the sample (APE, or average partial effects).

- **Partial effects for dummy variables should be computed as discrete differences in predicted values, not scaled coefficients (though the latter is often a surprisingly good approximation to the former).**

- **When there are nonlinearities in the index function of the model, such as $\mathbf{\beta}' \mathbf{x} = \beta_0 + \beta_1 z + \beta_2 z^2$, the program should compute a partial effect for $z$ using the chain rule, not meaningless scaled coefficients for $z$ and separately for $z^2$.**

- **When there are interactions in the index function model, such as $\mathbf{\beta}' \mathbf{x} = \beta_0 + \beta_1 Ed + \beta_2 Fem + \beta_3 Ed \times Fem$, the partial effects for $Ed$ and $Fem$ (or the interactions in general) should account for the interaction. There is no separate partial effect for the product term.**

- **Partial effects for the components of categorical variables can be analyzed in terms of transitions from one level to another, not always strictly between the categories and the base case.**

*NLOGIT*’s partial effects estimator, accessed with the command **PARTIALS**, accounts for all of these aspects. The basic command is

```
PARTIALS ; Effects : variable $
```

More than one variable can be analyzed by separating the names with slashes (not commas). An example appears in Figure 44. The model command is **PROBIT;Lhs=doctor;Rhs=one,age,educ,female,married,female*educ**

```
|-> PARTIALS ; Effects : age / educ / female / married ; Summary $
Partial Effects for Probit Probability Function
Partial Effects Averaged Over Observations
* --> Partial Effect for a Binary Variable

<table>
<thead>
<tr>
<th>Delta method</th>
<th>Effect</th>
<th>Error</th>
<th>t</th>
<th>95% Confidence Interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGE</td>
<td>-.00583</td>
<td>.00100</td>
<td>5.65</td>
<td>-.01067, -.00100</td>
</tr>
<tr>
<td>EDUC</td>
<td>-.01290</td>
<td>.00454</td>
<td>2.84</td>
<td>-.01210, -.01360</td>
</tr>
<tr>
<td>FEMALE</td>
<td>.12965</td>
<td>.02119</td>
<td>6.12</td>
<td>.08612, .17317</td>
</tr>
<tr>
<td>MARRIED</td>
<td>.04179</td>
<td>.02976</td>
<td>1.41</td>
<td>-.00477, .08836</td>
</tr>
</tbody>
</table>
```

**Figure 44. Partial Effects for a PROBIT Model**

Partial effects are computed by averaging across observations (average partial effects). Partial effects are computed at sample means by using ;Means.

A Tip: In a very large sample, average partial effects can take a very long time to compute. Use ;Means.

There are several ways to analyze scenarios with the variables in the model. The next example illustrates.

```
Example: LOGIT ; Lhs = Doctor
SIMULATE ; Rhs = one,age,educ,income,female,age*female $
PARTIALS ; Scenario : & age=25(2)65 ;plot(ci) $
```
<table>
<thead>
<tr>
<th>DOCTOR</th>
<th>Coefficient</th>
<th>Standard Error</th>
<th>z</th>
<th>Prob.</th>
<th>95% Confidence Interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constant</td>
<td>-0.29872</td>
<td>0.36477</td>
<td>-0.82</td>
<td>0.4128</td>
<td>(-1.05367, 0.41622)</td>
</tr>
<tr>
<td>AGE</td>
<td>-0.01620***</td>
<td>0.00485</td>
<td>3.35</td>
<td>0.0008</td>
<td>(-0.02486, -0.00754)</td>
</tr>
<tr>
<td>EDUC</td>
<td>-0.06580***</td>
<td>0.02055</td>
<td>-3.21</td>
<td>0.0013</td>
<td>(-0.10618, -0.02561)</td>
</tr>
<tr>
<td>INCOME</td>
<td>-0.07426</td>
<td>0.03968</td>
<td>-1.87</td>
<td>0.0618</td>
<td>(-0.15319, 0.0047)</td>
</tr>
<tr>
<td>FEMALE</td>
<td>-0.08732**</td>
<td>0.04218</td>
<td>-2.04</td>
<td>0.0415</td>
<td>(-0.17170, -0.00293)</td>
</tr>
</tbody>
</table>

Note: ***, **, * ==> Significance at 1%, 5%, 10% level.

Simulations are computed by average over sample observations.

<table>
<thead>
<tr>
<th>User Function</th>
<th>Function Value</th>
<th>Error</th>
<th>t</th>
<th>95% Confidence Interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age 25.00</td>
<td>0.51062</td>
<td>0.02273</td>
<td>22.48</td>
<td>0.46628, 0.55537</td>
</tr>
<tr>
<td>AGe 27.00</td>
<td>0.52361</td>
<td>0.02005</td>
<td>25.11</td>
<td>0.48274, 0.56448</td>
</tr>
<tr>
<td>AGe 29.00</td>
<td>0.53641</td>
<td>0.01830</td>
<td>28.19</td>
<td>0.50512, 0.56770</td>
</tr>
<tr>
<td>AGe 31.00</td>
<td>0.54920</td>
<td>0.01727</td>
<td>31.80</td>
<td>0.51536, 0.58305</td>
</tr>
<tr>
<td>AGe 33.00</td>
<td>0.56198</td>
<td>0.01562</td>
<td>35.99</td>
<td>0.52137, 0.59258</td>
</tr>
<tr>
<td>AGe 35.00</td>
<td>0.57471</td>
<td>0.01411</td>
<td>40.73</td>
<td>0.54705, 0.60236</td>
</tr>
<tr>
<td>AGe 37.00</td>
<td>0.58738</td>
<td>0.01280</td>
<td>45.07</td>
<td>0.55229, 0.62148</td>
</tr>
<tr>
<td>AGe 39.00</td>
<td>0.59998</td>
<td>0.01176</td>
<td>51.02</td>
<td>0.57063, 0.6303</td>
</tr>
<tr>
<td>AGe 41.00</td>
<td>0.61248</td>
<td>0.01104</td>
<td>55.46</td>
<td>0.59084, 0.63413</td>
</tr>
<tr>
<td>AGe 43.00</td>
<td>0.62467</td>
<td>0.01070</td>
<td>58.40</td>
<td>0.60390, 0.64564</td>
</tr>
<tr>
<td>AGe 45.00</td>
<td>0.63713</td>
<td>0.01074</td>
<td>59.30</td>
<td>0.61687, 0.65518</td>
</tr>
<tr>
<td>AGe 47.00</td>
<td>0.64926</td>
<td>0.01034</td>
<td>58.22</td>
<td>0.62790, 0.67107</td>
</tr>
<tr>
<td>AGe 49.00</td>
<td>0.66119</td>
<td>0.01003</td>
<td>55.90</td>
<td>0.63800, 0.68437</td>
</tr>
<tr>
<td>AGe 51.00</td>
<td>0.67296</td>
<td>0.01027</td>
<td>52.88</td>
<td>0.64802, 0.69790</td>
</tr>
<tr>
<td>AGe 53.00</td>
<td>0.68458</td>
<td>0.01036</td>
<td>49.74</td>
<td>0.65756, 0.71151</td>
</tr>
<tr>
<td>AGe 55.00</td>
<td>0.69551</td>
<td>0.01040</td>
<td>46.77</td>
<td>0.66675, 0.72508</td>
</tr>
<tr>
<td>AGe 57.00</td>
<td>0.70707</td>
<td>0.01063</td>
<td>44.10</td>
<td>0.67565, 0.73850</td>
</tr>
<tr>
<td>AGe 59.00</td>
<td>0.71800</td>
<td>0.01079</td>
<td>41.78</td>
<td>0.68432, 0.75189</td>
</tr>
<tr>
<td>AGe 61.00</td>
<td>0.72870</td>
<td>0.01132</td>
<td>38.78</td>
<td>0.69280, 0.76460</td>
</tr>
<tr>
<td>AGe 63.00</td>
<td>0.73935</td>
<td>0.01141</td>
<td>35.90</td>
<td>0.70111, 0.77718</td>
</tr>
<tr>
<td>AGe 65.00</td>
<td>0.74994</td>
<td>0.01144</td>
<td>33.66</td>
<td>0.70927, 0.78941</td>
</tr>
<tr>
<td>AGe 67.00</td>
<td>0.75928</td>
<td>0.01141</td>
<td>31.46</td>
<td>0.71730, 0.80125</td>
</tr>
</tbody>
</table>

Figure 45. Estimated Binary Logit Model and Simulation
Partial Effects Analysis for Logit Probability Function

Effects on function with respect to AOE
Partial effects for continuous AOE computed by differentiation
Effect is computed as derivative \( \frac{df}{dz} \)

| df/\(\delta\) AOE | Partial Effect | Standard Error | |t| | 95% Confidence Interval |
|-----------------|----------------|----------------|---------|------------------|
| AOE Function    | .00601         | .00098          | 6.15    | .00410           | .00793 |
| AOE 25.00       | .00639         | .00103          | 5.84    | .00424           | .00853 |
| AOE 27.00       | .00640         | .00110          | 5.81    | .00424           | .00856 |
| AOE 29.00       | .00640         | .00111          | 5.78    | .00423           | .00857 |
| AOE 31.00       | .00639         | .00111          | 5.76    | .00422           | .00857 |
| AOE 33.00       | .00638         | .00111          | 5.75    | .00421           | .00855 |
| AOE 35.00       | .00636         | .00109          | 5.74    | .00419           | .00852 |
| AOE 37.00       | .00632         | .00109          | 5.78    | .00418           | .00846 |
| AOE 39.00       | .00626         | .00108          | 5.81    | .00416           | .00839 |
| AOE 41.00       | .00622         | .00106          | 5.85    | .00414           | .00831 |
| AOE 43.00       | .00616         | .00104          | 5.92    | .00412           | .00820 |
| AOE 45.00       | .00609         | .00102          | 5.99    | .00410           | .00809 |
| AOE 47.00       | .00602         | .00099          | 6.09    | .00408           | .00795 |
| AOE 49.00       | .00593         | .00096          | 6.20    | .00405           | .00781 |
| AOE 51.00       | .00584         | .00092          | 6.34    | .00403           | .00764 |
| AOE 53.00       | .00574         | .00089          | 6.50    | .00401           | .00747 |
| AOE 55.00       | .00563         | .00084          | 6.60    | .00398           | .00729 |
| AOE 57.00       | .00552         | .00080          | 6.90    | .00395           | .00709 |
| AOE 59.00       | .00541         | .00076          | 7.14    | .00392           | .00689 |
| AOE 61.00       | .00529         | .00071          | 7.43    | .00389           | .00668 |
| AOE 63.00       | .00516         | .00067          | 7.76    | .00386           | .00647 |
| AOE 65.00       | .00503         | .00062          | 8.14    | .00382           | .00624 |
| AOE 67.00       | .00490         | .00057          | 8.50    | .00378           | .00602 |

Figure 46. Average Partial Effects over Scenario for Logit Model
4. Retained Results

The SIMULATE and PARTIALS instructions use the model estimates that are stored by the estimator. Several other results are stored for later use. Matrices B and VARB (the variance of the estimator) are stored as accessible matrices. The updated project window after the probit model in Figure 48 is estimated is shown in Figure 47. Note the appearance of the coefficient vector, the covariance matrix and the scalar log likelihood in Figure 30. The commands in Figure 31 test the hypothesis that the coefficients in the probit model are all zero using a Wald statistic. The statistic and the critical value from the chi squared table are shown in Figure 32.

Figure 47. Stored Matrix and Scalar Results

Figure 48. Matrix Manipulation

Figure 49. Using MATRIX and CALC to Carry Out a Test
C. Panel Data Forms

Nearly all of the models, such as REGRESS, PROBIT, LOGIT, POISSON, ORDERED, and so on. Generally, these are fixed effects, random effects, random parameters, and latent class models. (The last two of these are also usable with cross sections, but work well and naturally with panel data.) These have a variety of specifications and options all described in the program documentation. We list the basic forms here.

Panel data analysis begins with the SETPANEL instruction described in Section VI.E. The data must be arranged in contiguous blocks, by group. If your panel has 5,000 groups and 5 years of data on each group, the first 5 of the 25,000 rows of data are group 1, and so on. For the fixed and random effects models, the linear regression specification is different from all the other nonlinear specifications.

1. Fixed Effects Models

NOGIT’s fixed effects estimators are, with the exception of the binary logit model, unconditional estimators. The dummy variable coefficients are all computed. The limit on numbers of groups is hundreds of thousands. The binary logit model may be fit by the conditional (Chamberlain) estimator or the unconditional (Greene) estimator. The linear fixed effects regression is requested with

\[
\text{REGRESS} \quad ; \ Lhs = \ldots ; Rhs = \ldots ; \text{Panel} ; \text{FixedEffects} \$
\]

The general form for nonlinear models is

\[
\text{Model} \quad ; \ Lhs = \ldots ; Rhs = \ldots ; \text{Panel} ; \text{FEM} \$
\]

Two models must be estimated immediately prior in cross section form, FRONTIER and NEGBIN. E.g.,

\[
\text{FRONTIER} \quad ; \ Lhs = \ldots ; Rhs = \ldots \$
\]

\[
\text{FRONTIER} \quad ; \ Lhs = \ldots ; Rhs = \ldots ; \text{Panel} ; \text{FEM} \$
\]

The negative binomial looks the same. The other three panel data forms, random effects, random parameters and latent class models for the stochastic frontier and negative binomial models are estimated the same way. There are also a large number of other panel data specification for the stochastic frontier model.

There is a distinction for the logit model.

\[
\text{LOGIT} \quad ; \ldots ; \text{Panel} ; \text{FEM} \$ \text{ is for the unconditional estimator}
\]

\[
\text{LOGIT} \quad ; \ldots ; \text{Panel} ; \text{FIXED} \$ \text{ requests the conditional (Chamberlain) estimator.}
\]
**Figure 50. Linear Fixed Effects Model**

```
<table>
<thead>
<tr>
<th>Variable =</th>
<th>Variable Groups</th>
<th>Max</th>
<th>Min</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>TI</td>
<td>Group sizes</td>
<td>1</td>
<td>550</td>
<td>7</td>
</tr>
<tr>
<td>Frequency count for group sizes of TI</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Group size = 1</td>
<td>Pct = 20.36%</td>
<td>CumPct = 20.36%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Group size = 2</td>
<td>Pct = 12.55%</td>
<td>CumPct = 32.91%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Group size = 3</td>
<td>Pct = 12.36%</td>
<td>CumPct = 45.27%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Group size = 4</td>
<td>Pct = 17.64%</td>
<td>CumPct = 63.91%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Group size = 5</td>
<td>Pct = 14.55%</td>
<td>CumPct = 77.45%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Group size = 6</td>
<td>Pct = 12.91%</td>
<td>CumPct = 90.36%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Group size = 7</td>
<td>Pct = 9.64%</td>
<td>CumPct = 100.00%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Ordinary least squares regression  

```
<table>
<thead>
<tr>
<th>LHS</th>
<th>INCOME</th>
<th>Mean</th>
<th>34350</th>
</tr>
</thead>
<tbody>
<tr>
<td>Std. deviation</td>
<td>15.21</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No. of observations</td>
<td>2039</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sum of squares</td>
<td>7.6005</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Residual</td>
<td>48.2400</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>56.2322</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Standard error of e</td>
<td>1.5298</td>
<td></td>
<td></td>
</tr>
<tr>
<td>fit B-squared</td>
<td>0.1419</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prob F &gt; F*</td>
<td>0.0000</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Panel Data Analysis of INCOME (one way)  

```
<table>
<thead>
<tr>
<th>Source</th>
<th>Variation</th>
<th>Degree of Freedom</th>
<th>Mean Square</th>
</tr>
</thead>
<tbody>
<tr>
<td>Between</td>
<td>42.9472</td>
<td>54</td>
<td>.0782</td>
</tr>
<tr>
<td>Residual</td>
<td>13.2049</td>
<td>148</td>
<td>.0093</td>
</tr>
<tr>
<td>Total</td>
<td>56.2322</td>
<td>154</td>
<td>.0275</td>
</tr>
</tbody>
</table>

```

Panel: Groups Empty  

```
| Input | Standard Error | z | Prob (>|z|) | 95% Confidence Interval |
|-------|----------------|---|------------|-------------------------|
| .08001 | .00895 | -9.35 | .0000 |
| .01064 | .00897 | 1.20 | .1176 |
| .00953 | .00897 | 1.09 | .2935 |
| .01059 | .00910 | 1.18 | .1220 |
| Constant | .03630 | .0240 | 1.50 | .01131 |

Note: ***,**, * --> Significance at 1%, 5%, 10% level.

```

Panel: Groups Empty  

```
| Input | Standard Error | z | Prob (>|z|) | 95% Confidence Interval |
|-------|----------------|---|------------|-------------------------|
| .559 | .0559 | -.10 | .9176 |
| .01326 | .00872 | 1.53 | .0620 |
| .01092 | .00872 | 1.26 | .1046 |
| .00959 | .00984 | 1.00 | .2935 |

Note: ***,**, * --> Significance at 1%, 5%, 10% level.

```

42
2. Random Effects Models

Several models, including REGRESS, PROBIT, LOGIT, ORDERED, POISSON and NEGBIN support familiar random effects forms. About 50 models provide a random parameters form, so all of those allow a random effects model in the form of a random constant term model. For the first set, the form of the command is the same for the linear and nonlinear models,

\[
\text{Model ; Lhs = \ldots ; Rhs = \ldots ; Panel ; Random Effects} \]

3. Random Parameters Models

A random parameters model is defined by defining the model, then defining which parameters are random. The model is estimated by maximum simulated likelihood. Some additional settings may be made to control the simulation.

\[
\text{Model ; Lhs = dependent variable} \\
\text{; Rhs = one,\var1,\var2,\ldots,\varK (list of variables, usually including one)} \\
\text{; RPM ; Panel} \\
\text{; Fcn = var(n), \ldots, var(n)} \]

where ‘\var’ is a name of a variable that appears in the Rhs list. The simulation can be based on random draws or preferably on Halton sequences which produce better results. An example of a model with six regressors, two random parameters, appears in Figure 35. In the example, the command is

\[
\text{PROBIT ; Lhs = doctor} \\
\text{; Rhs = one,age,educ,married,female,hhkids} \\
\text{; RPM ; Panel} \\
\text{; Fcn = one(n),female(n)} \\
\text{; Halton ; Draws = 50} \]

To specify this as a simple random effects model, we would change the function definition to \text{; Fcn=one(n)}. In the specifications above, the ‘(n)’ indicates a normally distributed parameter. There are 15 other distributions that can be used. An important feature of the RP models is the conditional estimates of the random parameters, E[\beta_i|data_i]. This is requested with \text{; Parameters} and creates a matrix named BETA_I that can be further analyzed.
Figure 52. Random Parameters Model Command and Results

4. Latent Class Models

A latent class model is specified with

\[
\text{Model} \; ; \; \text{Lhs} = \text{dependent variable} \\
\text{Rhs} = \text{one, var1, var2,..., varK} \; (\text{list of variables, usually including one}) \\
\text{LCM} ; \; \text{Panel} \\
\text{Pts} = \text{number of classes}
\]

There are a variety of forms of LC models. It is possible to impose constraints across classes to create many different types of models.

Example:

\[
\text{LOGIT} \; ; \; \text{Lhs} = \text{Doctor} \\
\text{Rhs} = \text{one, age, educ, income, female} \\
\text{Panel} ; \; \text{LCM} ; \; \text{Pts} = 2 \\
\text{Parameters}
\]

In the example, we fit a two class latent class binary logit model. The :Parameters requests computation of a matrix of conditional class probabilities. The estimated model, updated project window and 18 of the 550 rows of the class probabilities matrix are displayed in Figure 53.
Figure 53. Latent Class Binary Logit Model
VIII. Multinomial Logit and Multinomial Choice

*NLOGIT* contains all of *LIMDEP* plus an additional set of model estimators and analysis tools for multinomial choice models such as the multinomial logit and multinomial probit specifications. The canonical form of the model is illustrated with this example that appears in our sample data set. A model for four models of travel, mode \( \in \{\text{Air, Train, Bus, Car}\} \) defines the probability that an individual will choose one of the four. The underlying model is a random utility specification for individual \( i \) and modes \( 1,\ldots,J \):

\[
U_{i,\text{mode}} = \alpha_{\text{mode}} + \beta_{\text{time}} \text{TIME}_{i,\text{mode}} + \beta_{\text{cost}} \text{COST}_{i,\text{mode}} + \gamma_{\text{mode}} \text{INCOME}_i + \varepsilon_{i,\text{mode}} \\
Y_{i,\text{mode}} = 1[U_{i,\text{mode}} > \max_{j \neq i} U_{i,\text{mode}}] \text{ (Y}_{i,\text{mode}} \text{ equals 1 for the mode with maximum utility, 0 else.)} \\
\varepsilon_{i,\text{mode}} \sim \text{Type I extreme value, independent across } i \text{ and mode.}
\]

The specification implies that

\[
\text{Prob}(Y_{i,\text{mode}} = 1) = \frac{\exp(\alpha_{\text{mode}} + \beta_{\text{time}} \text{TIME}_{i,\text{mode}} + \beta_{\text{cost}} \text{COST}_{i,\text{mode}} + \gamma_{\text{mode}} \text{INCOME}_i)}{\sum_{\text{modes}} \exp(\alpha_{\text{mode}} + \beta_{\text{time}} \text{TIME}_{i,\text{mode}} + \beta_{\text{cost}} \text{COST}_{i,\text{mode}} + \gamma_{\text{mode}} \text{INCOME}_i)}
\]

This is the basic multinomial logit model. (Notice that the specification involves variables (TIME, COST) that vary across choices and a variable (INCOME) that does not vary across the choices. It is not necessary to distinguish. Mathematically, it is necessary to normalize the coefficients so that one of the \( \alpha_{\text{mode}} \) parameters and one of the \( \gamma_{\text{mode}} \) parameters equals zero.) This is the basic model for multinomial choice. *NLOGIT* provides this model, a large number of extensions of the specification, such as the multinomial probit and nested logit models, and a set of analysis tools (similar to *SIMULATE* and *PARTIALS*).

A Tip: The CLOGIT command in *LIMDEP* is provided for the basic multinomial logit model. The extensions (as well as CLOGIT) are provided by *NLOGIT*.

A. Data

The data for this part of the description of *NLOGIT* are contained in the CSV file, mnc.csv (‘mnc’ for ‘multinomial choice’). To replicate the examples and learn how to fit the models, you should IMPORT this file. There is also a project file provided, mnc.lpj, which you can LOAD directly. This data file contains 12,800 observations in two data sets. The first data set contains 12 variables (columns), the second contains 7 – they are arranged side by side in 20 columns. The first 12 are 12,800 observations equal to 400 individuals times 8 repetitions (it is a panel) times 4 choices. The second data set contains 840 observations equal to 210 individuals times 4 choices in each observation. The 840 observations appear in the first 840 rows of their part of the data set. The rows below them (841-12800) contain missing values for these 8 variables. The shorter data set applies to the travel mode example described above.

An Important Tip: When you enter the data for multinomial choice analysis, the IMPORT step does not account for the internal structure of the data set. Our file, mnc.csv, is imported simply as 12,800 rows of data. Like a panel data set, the internal structure of the data is accounted for when the data are used to fit a model.

Data for multinomial choice modeling resemble a panel data set. The data set is arranged in blocks of data for each person for each choice situation. Our examples both describe choices over 4 alternatives. The data are thus arranged with a line of data for each alternative in the choice set for the person. This is indicated in Figure 54.

A Tip: It is possible to work with choice data arranged on a single line – what some other programs call the ‘wide form.’ This is extremely cumbersome and greatly limits the range of specifications and model sizes. *NLOGIT* does provide a way to use these data, and to convert them to the more accommodating ‘long form.’

A Second Tip: *NLOGIT* allows the number of choices in the choice set to vary across individuals. Our first data set is a choice experiment that has 8 choice situations for each person. *NLOGIT* also allows the number of choice situations in a stated choice data set to vary across individuals.
The data in Figure 54 are 210 observations on four travel modes, AIR, TRAIN, BUS, CAR in the respective 4 rows. Notice that the first variable, MODE, is $Y_{i,\text{mode}}$ in our mathematical example. The first four individuals in the sample all chose AIR, as the 4th row equals one in each case. There are several variables that vary across the choices – they are the attributes: TTME = terminal time (waiting time to begin the journey), INVC = in-vehicle cost, INVT = in-vehicle time, GC = a generalized cost measure. There are also two variables that do not vary across choices, HINC = household income and PSIZE = party size. These are characteristics of the person (traveler). It is not necessary to expand choice invariant variables. This is done internally as part of the model specification.

B. Basic Multinomial Choice Model and Choice Substitution Elasticities

The essential command for a multinomial logit model is

\begin{verbatim}
CLOGIT ; Choices = list of names for the choices
; Lhs = the choice variable
; Rhs = attributes that vary across the choices
; Rh2 = characteristics that do not vary across choices
\end{verbatim}

Figure 55 illustrates. Note, if you include ONE in your Rhs list, it is automatically moved to the Rh2 list. Models can be specified with either or both Rhs or Rh2 variables. Neither is required. If you do not have an Rh2 list, but you include ONE on your Rhs, the program creates an Rh2 list for you and puts ONE in it. This is not done for any other variables.

Figure 55 shows the estimation results for the commands in Figure 55. This is the standard form of the display for the multinomial logit model. The next section lists some of the different choice models that can be specified. Estimation of every choice model begins with a starting values step at which the basic multinomial logit model is fit.
One of the major functions of the estimated choice model is to provide estimates of the impact of changes in relevant variables on the substitution patterns among the alternatives. Choice elasticities are the common device for this computation. The elasticity is defined as

\[ E: \text{Attribute}(\text{choice}) = \text{The effect on the probabilities of the choices when attribute in a particular choice changes.} \]

For example, \( E: \text{cost}(\text{air}) \) is the effect of changes in the cost of air on the probabilities of choosing the alternatives. Each attribute in the model produces a full matrix of elasticities. Elasticities are requested with the specification:

\[ \text{Effects: attribute (alternatives desired)} \]

It is common to request the effect of a change in an attribute in all choices. The following example shows how to do this. The '*' means 'all alternatives.'

\[
\text{NLOGIT ; Choices = air,train,bus,car} \\
\quad ; \text{Lhs = mode} \\
\quad ; \text{Rhs = invt, invc} \\
\quad ; \text{Rh2 = one, hinc} \\
\quad ; \text{Effects: invc(*)} \$
\]

This produces the table shown in Figure 57. This is the effect of changes in INVC on the probabilities of all all alternatives. If the specification had been invc(air,train), then only the first two rows of the table would be shown.

\[
\begin{array}{|c|c|c|c|c|}
\hline
\text{INVC} & \text{AIR} & \text{TRAIN} & \text{BUS} & \text{CAR} \\
\hline
\text{AIR} & -5.115 & 2.196 & 2.196 & 2.196 \\
\text{TRAIN} & 1.040 & -3.363 & 1.040 & 1.040 \\
\text{BUS} & 0.392 & 0.392 & -0.247 & 0.392 \\
\text{CAR} & 0.0437 & 0.0437 & 0.0437 & -0.1363 \\
\hline
\end{array}
\]

Figure 57. Estimated Choice Elasticities
The table of elasticities can be expanded to include much more information by adding

; Full

to the command. This produces the results such as shown in Figure 58.

| Choice | Coefficient | Standard Error | z | Prob | | 95% Confidence Interval |
|--------|-------------|----------------|---|------|----------------------------|
| AIR    | -51.156***  | .01126         | -45.45 | .0000 | -53.355 - 48.944          |
| TRAIN  | 2186.0***   | .01160         | 18.93  | .0000 | 1568.8 - 2423.4           |
| BUS    | 2196.0***   | .01160         | 18.93  | .0000 | 1568.8 - 2423.4           |
| CAR    | 2196.0***   | .01160         | 18.93  | .0000 | 1568.8 - 2423.4           |

| Choice | Coefficient | Standard Error | z | Prob | | 95% Confidence Interval |
|--------|-------------|----------------|---|------|----------------------------|
| AIR    | 1040.0***   | .00440         | 23.63  | .0000 | 9537.8 - 11253.2          |
| TRAIN  | 3362.6***   | .01583         | -21.24 | .0000 | 3472.5 - 3252.4          |
| BUS    | 1040.0***   | .00440         | 23.63  | .0000 | 9537.8 - 11253.2          |
| CAR    | 1040.0***   | .00440         | 23.63  | .0000 | 9537.8 - 11253.2          |

| Choice | Coefficient | Standard Error | z | Prob | | 95% Confidence Interval |
|--------|-------------|----------------|---|------|----------------------------|
| AIR    | 0.3918***   | .00135         | 29.00  | .0000 | 0.3653 - 0.4182           |
| TRAIN  | 0.3918***   | .00135         | 29.00  | .0000 | 0.3653 - 0.4182           |
| BUS    | 2.4734***   | .00681         | 36.35  | .0000 | 2.3610 - 2.5859           |
| CAR    | 0.3918***   | .00135         | 29.00  | .0000 | 0.3653 - 0.4182           |

| Choice | Coefficient | Standard Error | z | Prob | | 95% Confidence Interval |
|--------|-------------|----------------|---|------|----------------------------|
| AIR    | 0.4371***   | .00202         | 21.63  | .0000 | 0.3975 - 0.4767           |
| TRAIN  | 0.4371***   | .00202         | 21.63  | .0000 | 0.3975 - 0.4767           |
| BUS    | 0.4371***   | .00202         | 21.63  | .0000 | 0.3975 - 0.4767           |
| CAR    | -1363.4***  | .00752         | -18.12 | .0000 | -1510.9 - -1215.9         |

Figure 58. Full Display of Results for Elasticities
C. Multinomial Choice Models

Most of the extensions of the multinomial logit model are requested by modifying the basic command. The following will list a few of these by way of extending the example in Figures 55 and 56.

1. Multinomial Probit Model

The multinomial probit (MNP) model is an extension of the logit model. The MNP model allows some heteroscedasticity across choices as well as correlation of the utility functions. This is the usual first extension of the MNL model to relax the independence from irrelevant alternatives (IIA) assumptions. The model is requested simply by adding ;MNP to the basic specification. Since it is a simulation based estimator, sometimes it is a good idea to control the number of draws, as shown here.

```
NLOGIT ; Choices = air,train,bus,car
; Lhs = mode
; Rhs = invt, invc
; Rh2 = one, hinc
; MNP ; Draws = 5 ; Maxit = 5 $
```

(The command has used a very small number of draws and only 5 iterations. This estimator takes a very large amount of time. The results below show the results with 10 draws and allowing it to reach convergence.)

```
Multinomial Probit Model
Dependent variable MODE
Log likelihood function -222.31250
Restricted log likelihood -291.22322
Lhs squared (. 1.9 .1.1) 157.51865
Significance level 0.00000
Moaddes Pseudo R-squared 2563482
Estimation based on N = 210, K = 15
Inf.Cr. AIC = 470.6 AIC/N = 2.241
Model estimated: Apr 21, 2013, 19:42:32
S+1-LogLik/LogLik Log-1 from R-sqrd RAdj
No coefficients -291.22322 2264 2203
Constants only -283.75989 2165 2000
At start values -214.4449 1260 1079
Response data are given as ind. choices
Replications for simulated probe = 210
Used pseudo random draws (Reference twister)
Number of obs = 210. skipped 0 obs

<table>
<thead>
<tr>
<th>MODE</th>
<th>Coefficient</th>
<th>Standard Error</th>
<th>z</th>
<th>Prob</th>
<th>% Confidence Interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVt</td>
<td>-0.00611**</td>
<td>.00130</td>
<td>-7.94</td>
<td>0.000</td>
<td>-0.0074 - 0.0048</td>
</tr>
<tr>
<td>INVc</td>
<td>-0.03970***</td>
<td>.00911</td>
<td>-4.40</td>
<td>0.000</td>
<td>-0.0569 - 0.0217</td>
</tr>
<tr>
<td>AIR</td>
<td>-3.27626**</td>
<td>1.32235</td>
<td>-2.49</td>
<td>0.014</td>
<td>-5.07522 - 0.4840</td>
</tr>
<tr>
<td>BVRN</td>
<td>0.32722**</td>
<td>0.27255</td>
<td>1.20</td>
<td>0.230</td>
<td>-0.15417 - 0.6045</td>
</tr>
<tr>
<td>TRA</td>
<td>0.02215**</td>
<td>0.01251</td>
<td>1.79</td>
<td>0.074</td>
<td>0.0010 3.0258</td>
</tr>
<tr>
<td>BUS</td>
<td>-0.0510**</td>
<td>0.01637</td>
<td>-3.15</td>
<td>0.002</td>
<td>-0.08260 - 0.0080</td>
</tr>
</tbody>
</table>

SDs. Dev. of the Normal Distribution
s[AIR] 3.08329*** 1.06600 2.78 0.002 1.02477 5.77521
s[TRAV] 0.41834* 0.45467 1.85 0.064 0.15010 1.73210
s[CAR] 1.0 (Fixed Parameter) ...

Correlations in the Normal Distribution
r[AIR,TRAV] 0.35846 0.73381 0.46 0.4825 -1.1016 1.77173
r[AIR,BUS] 0.51379 0.5134 1.08 0.2806 0.46205 1.56208
r[TRAV,BUS] 0.78151*** 0.9859 0.96 0.0001 0.39781 1.57872
r[TRAV, CAR] 0.0 (Fixed Parameter) ...
```

Figure 59. Estimated Multinomial Probit Model
2. Nested Logit Model

NLOGIT allows up to 4 levels in a nested logit model. A nested logit model is specified simply by providing the tree structure in the NLOGIT command.

\[
\text{NLOGIT } \begin{cases} \text{Choices} = \text{air, train, bus, car} \\
\text{Lhs} = \text{mode} \\
\text{Rhs} = \text{invt, invc} \\
\text{Rh2} = \text{one, hinc} \\
\text{Tree} = \text{Private(air, car), public(train, bus)} \end{cases}
\]

Tables of elasticities for a nested logit model include a decomposition of the total effect of switching between branches and substitution within a branch.

```
FINL Nested Multinomial Logit Model
Dependent variable MODE
Log likelihood function -223.84995
Restricted log likelihood -294.12382
Chi squared [ 10 d.f.] 154.64379
Significance level 0.05000
Excluded Pseudo R-squared 0.07651
Estimation based on N = 210 & K = 10
Int Cc AIC = 467.7 AIC = 2.437
R^2-Less/Log= Log-Less R-squared RLSQ:
Rs coefficients = -291.1210 .2311 .2187
Constants only -285.7950 .2111 .2784
At start values -249.2565 .1919 .6974
Response data are given as ind. choices
The model has 2 levels:
  Nested Logit form: V(pars)=Tab[r].r.Blk[1]r
  & Pr No normalizations imposed a priori
Number of obs = 210. skipped 0 obs

<table>
<thead>
<tr>
<th>Mode</th>
<th>Coefficient</th>
<th>Standard Error</th>
<th>Prob</th>
<th>90% Confidence Interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>INV1</td>
<td>-0.0079***</td>
<td>0.0055</td>
<td>-0.07</td>
<td>-0.0245</td>
</tr>
<tr>
<td>INV2</td>
<td>-0.0064***</td>
<td>0.0055</td>
<td>-0.07</td>
<td>-0.0245</td>
</tr>
<tr>
<td>AIR</td>
<td>-1.2946****</td>
<td>0.0210</td>
<td>-1.00</td>
<td>-1.0029</td>
</tr>
<tr>
<td>AIR_HH</td>
<td>0.0072</td>
<td>0.0045</td>
<td>0.00</td>
<td>0.0071</td>
</tr>
<tr>
<td>L1_HH</td>
<td>0.0367</td>
<td>0.0214</td>
<td>0.02</td>
<td>0.0214</td>
</tr>
<tr>
<td>TRA_HH</td>
<td>-0.0080</td>
<td>0.0055</td>
<td>-0.02</td>
<td>-0.0143</td>
</tr>
<tr>
<td>BUS</td>
<td>-0.0065***</td>
<td>0.0055</td>
<td>-0.07</td>
<td>-0.0245</td>
</tr>
<tr>
<td>BUS_HH</td>
<td>0.0011</td>
<td>0.0045</td>
<td>0.00</td>
<td>0.0011</td>
</tr>
<tr>
<td>IV parameters: ten[hl][r], sig[al][r], ph[r]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PRIVATE</td>
<td>12.5477***</td>
<td>3.22704</td>
<td>3.96</td>
<td>3.22702</td>
</tr>
<tr>
<td>PUBLIC</td>
<td>7.73324***</td>
<td>1.96806</td>
<td>3.96</td>
<td>1.96804</td>
</tr>
</tbody>
</table>
```

Figure 58. Estimated Nested Logit Model

```
Attribute is INV in choice AIR
Decomposition of Effect of Nest Branch Choice Total Effect
Trunk=Trunk(1)
Branch=PRIVATE
* Choice=AIR .000 .000 .157 .026 -.182 .011
Choice=CAR .000 .000 .157 .026 -.182 .011
Branch=PUBLIC
Choice=TRAIN .000 .000 .184 .000 .184 .011
Choice=BUS .000 .000 .184 .000 .184 .011
```

Elasticity wrt change of X in row choice on Prob[colume choice]

```
<table>
<thead>
<tr>
<th>INV</th>
<th>AIR</th>
<th>CAR</th>
<th>TRAIN</th>
<th>BUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIR</td>
<td>-1.021</td>
<td>-1.299</td>
<td>.1840</td>
<td>.1840</td>
</tr>
<tr>
<td>CAR</td>
<td>.0252</td>
<td>.0396</td>
<td>.0391</td>
<td>.0391</td>
</tr>
<tr>
<td>TRAIN</td>
<td>.0640</td>
<td>.0640</td>
<td>-.1035</td>
<td>-.0721</td>
</tr>
<tr>
<td>BUS</td>
<td>.0216</td>
<td>.0215</td>
<td>-.0252</td>
<td>-.0457</td>
</tr>
</tbody>
</table>
```

Figure 59. Estimated Elasticities for a Nested Logit Model
3. Mixed (Random Parameters, RP) Logit Model and Willingness to Pay (WTP)

The mixed (random parameters) logit model is the platform for the most recent, advanced formulations of the multinomial choice models in NLOGIT. The RP logit model is specified by providing the definition of the random parameters and, if desired, controls for the simulations.

Willingness to pay (WTP) is often measured in a choice model. The typical calculation is based on

\[
WTP_{attribute} = \frac{\beta_{attribute}}{\beta_{income}}
\]

Which measures the marginal utility of the attribute divided by the marginal utility of income. When income does not appear in the model, often the negative of a cost coefficient is used as a proxy for the marginal utility of income. When the model has fixed (nonrandom) coefficients, the WTP can be computed simply as the ratio of two coefficients (with a calculator). When parameters are random, WTP will vary across individuals if either of the components does. Figures 60 and 61 show estimation of a random parameters model and examination of the estimates of WTP.

\[
\text{RPLOGIT ; Lhs=mode ; Choices=air,train,bus,car} \\
\text{Rhs=invn,invc} \\
\text{Rhs2=one,invc} \\
\text{Pts=50 ; Halton} \\
\text{Fcn=invn(n) ; This specifies a single random parameter.} \\
\text{This can be expanded, e.g., invn(n), invc(n).} \\
\text{Wtp=invn/invc ; Parameters} \\
\text{KERNEL} \\
\text{Rhs=wtp_i} \\
\text{Title=Estimated Distribution of WTP Across Sample}
\]

<table>
<thead>
<tr>
<th>Random Parameters Logit Model</th>
<th>Standard Error</th>
<th>z</th>
<th>Prob</th>
<th>90% Confidence Interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dependent variable</td>
<td>N = 210, K = 9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Restricted log likelihood</td>
<td>-288.7888</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chi squared (d.f.)</td>
<td>524.3 3185</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Significance level</td>
<td>0.0000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>McFadden Pseudo R-squared</td>
<td>0.243 1904</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Estimation based on N = 210, K = 9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model estimated</td>
<td>Apr 11, 2012, 22:05:22</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rsq-1.Log/Logit Log-T (Form) R-squared RAdj</td>
<td>0.366 2968</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rsq constant only</td>
<td>-288.7888 0.366 2968</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rsq start values</td>
<td>-240.2655 0.366 2968</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rsq response data given as ind. choices</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Missing Halton sequence in simulations</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HMR estimator used for asymptotic variance</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Numb of obs. - 210, skipped 6 obs</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Random parameters in utility functions</td>
<td>68054 -12.35 0.0800 -12.253 -0.0800</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nonrandom parameters in utility functions</td>
<td>63754 -12.35 0.0800 -12.253 -0.0800</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INV</td>
<td>62983 -12.35 0.0800 -12.253 -0.0800</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Train</td>
<td>63754 -12.35 0.0800 -12.253 -0.0800</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A_Loc</td>
<td>62983 -12.35 0.0800 -12.253 -0.0800</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A_RLNG</td>
<td>60968 -12.35 0.0800 -12.253 -0.0800</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A_RNG</td>
<td>60968 -12.35 0.0800 -12.253 -0.0800</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A_RNG2</td>
<td>60968 -12.35 0.0800 -12.253 -0.0800</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A_RNG3</td>
<td>60968 -12.35 0.0800 -12.253 -0.0800</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A_RNG4</td>
<td>60968 -12.35 0.0800 -12.253 -0.0800</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A_RNG5</td>
<td>60968 -12.35 0.0800 -12.253 -0.0800</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Distance (km)</td>
<td>63754 11.46 0.0800 60968 11.46 0.0800</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Note: *** ** * = significant at 1%, 5%, 10% level</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 60 Estimated Random Parameters Model
D. Stated Choice (Panel) Data

Stated choice experiments are analogous to panel data. The individuals in the sample are observed several times. Our experimental data in mnc.csv consist of 400 individuals each observed making one of four choices, eight times. There are 32 rows of data for each individual. The first individual is shown in Figure 62. For purposes of specifying multinomial choice models that use this structure of the data, this panel has $Pds = 8$, not 32.

A Tip: Do not use ;setpanel to set up stated choice data. The count variable must be constructed appropriately by you. If the number of repetitions is fixed, you will be able to use ;Pds=Nrep in your command. You will not use ;Panel. In our models using these data, we will use ;Pds=8.

A Second Tip: These data are an ‘unlabeled’ choice set. The brands are distinguished only by their position in a list of brands. It is difficult to interpret substitution patterns in a model for choice with unlabeled alternatives.
Stated choice data allow the specification of essentially panel data models. The random utility models are variations on the general form

$$U(\text{choice})_{i,t,\text{mode}} = \beta_i'x_{i,t,\text{mode}} + \epsilon_{i,t,\text{mode}} + w_{i,\text{mode}}$$

$$\beta_i = \beta + u_i$$

The definition of $\beta_i$ implies that the parameters are random across individuals, but constant across choice situations. The random terms $w_{i,\text{mode}}$ are likewise constant across choice situations, and can be viewed as random effects. The constancy of the random terms in the model allows observations to be correlated within the group, which is the essential feature of panel data. (There are many variations on this model described in the manual.)

The stated choice data consist for each person of 8 repetitions on the choice of one of 3 brands or none of the above. The attributes are ‘fashion,’ ‘quality,’ ‘price’ and ‘price^2’. There are also two characteristics, gender coded as male=1 and female=0 and age, coded as a category for three brackets, under 25, 25-39, 40+.

1. Random Parameters Model

Figures 63 – 65 show estimation of a random parameters model with one random coefficient.

---

Figure 63 Command for a Mixed Logit Model

```
SAMPLE \ All 8
RPLOGIT ; LHS = Choice ; Choices = Brand1.Brand2.Brand3,None
; Rhm = -asc4,fash,qual,price
; Rh2 = male
; Drome = 50 ; Halton ; Pds = 6
; Fcu = price^2 ; $5
```

---

Start values obtained using MNL model

Dependent variable  Choice
Log likelihood function  -4145.28384
Estimation based on N = 1200, K = 7
Inf Cr AIC = 8904.6 AIC/W = 2.896
Model estimated: Apr 21, 2011, 18:53:22
R^2=1-Log-Like Log-L (from R-sqrd R logistic)
Constants only -2061.5882  0.0068  0.0069
Response data are given as ind choices
Number of obs = 1200, skipped 6 obs

<table>
<thead>
<tr>
<th></th>
<th>Coefficient</th>
<th>Standard Error</th>
<th>z</th>
<th>Prob [z&gt;z]</th>
<th>95% Confidence Interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>price</td>
<td>-12.925***</td>
<td>0.8878</td>
<td>-14.16 0.0000</td>
<td>-14.2228 -11.9291</td>
<td></td>
</tr>
<tr>
<td>ASC4</td>
<td>0.0956</td>
<td>0.0070</td>
<td>1.36 0.1724</td>
<td>-5.1922  6.2834</td>
<td></td>
</tr>
<tr>
<td>FASH</td>
<td>1.2550***</td>
<td>0.0736</td>
<td>17.37 0.0000</td>
<td>1.1151 1.3950</td>
<td></td>
</tr>
<tr>
<td>ZHAL</td>
<td>1.0359***</td>
<td>0.0497</td>
<td>20.85 0.0000</td>
<td>0.9374 1.1344</td>
<td></td>
</tr>
<tr>
<td>BRA_MAL1</td>
<td>-3.962***</td>
<td>0.0927</td>
<td>-43.60 0.0000</td>
<td>-4.1217 -3.8027</td>
<td></td>
</tr>
<tr>
<td>BRA_MAL2</td>
<td>0.0456</td>
<td>0.0044</td>
<td>10.98 0.0000</td>
<td>0.0438 0.0474</td>
<td></td>
</tr>
<tr>
<td>BRA_MAL3</td>
<td>0.0891</td>
<td>0.0045</td>
<td>19.62 0.0000</td>
<td>0.0842 0.0940</td>
<td></td>
</tr>
</tbody>
</table>

Note: *** ** * ++ Significance at 1%, 5%, 10% level

Figure 64. Multinomial Logit Starting Values
2. Error Components (Random Effects) Logit Model

*NLOGIT*’s Error Components Logit (ECLOGIT) model is equivalent to a random effects model. It is also possible to specify the logical equivalent of a nested logit model. The example below specifies a nested effects model in which one branch contains the three brands and a second contains the outside alternative, none. Figure 66 shows estimates of an error components model. Note how the error components are specified. The (brand1,brand2,brand3) specifies that the same effect appears in all three utility functions.

```
ECLOGIT ; LHS = Choice ; Choices = Brand1,Brand2,Brand3,None
; Rhs = asc4,fash,qual,price
; Rh2 = male
; Draws = 50 ; Halton ; Pds = 8
; Ecm=(brand1,brand2,brand3),(none)$
```

### Table 1: Random Parameters in Utility Functions

<table>
<thead>
<tr>
<th>Variable</th>
<th>Coefficient</th>
<th>Standard Error</th>
<th>z</th>
<th>Prob</th>
<th>95% Confidence Interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>price</td>
<td>-2.320***</td>
<td>0.110</td>
<td>-21.0</td>
<td>&lt;0.001</td>
<td>-2.539 to -2.102</td>
</tr>
</tbody>
</table>

### Table 2: Nonrandom Parameters in Utility functions

<table>
<thead>
<tr>
<th>Variable</th>
<th>Coefficient</th>
<th>Standard Error</th>
<th>z</th>
<th>Prob</th>
<th>95% Confidence Interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASC4</td>
<td>-0.0753</td>
<td>0.010</td>
<td>-7.5</td>
<td>0.000</td>
<td>-0.100 to -0.050</td>
</tr>
<tr>
<td>FASH</td>
<td>1.582***</td>
<td>0.072</td>
<td>22.1</td>
<td>0.000</td>
<td>1.439 to 1.725</td>
</tr>
<tr>
<td>QUAL1</td>
<td>-1.025***</td>
<td>0.066</td>
<td>-15.5</td>
<td>0.000</td>
<td>-1.151 to -0.899</td>
</tr>
<tr>
<td>QUAL2</td>
<td>-1.025***</td>
<td>0.066</td>
<td>-15.5</td>
<td>0.000</td>
<td>-1.151 to -0.899</td>
</tr>
<tr>
<td>ERA_MAL1</td>
<td>-0.927**</td>
<td>0.072</td>
<td>-13.0</td>
<td>0.000</td>
<td>-1.067 to -0.787</td>
</tr>
<tr>
<td>ERA_MAL2</td>
<td>0.446</td>
<td>0.030</td>
<td>14.9</td>
<td>0.000</td>
<td>0.387 to 0.504</td>
</tr>
<tr>
<td>Ecm</td>
<td>-0.140</td>
<td>0.020</td>
<td>-7.0</td>
<td>0.000</td>
<td>-0.180 to -0.099</td>
</tr>
</tbody>
</table>

Note: *** , ** , * : Significance at 1%, 5%, 10% level.

Figure 65. Estimated Mixed Logit Model

Figure 66. Estimated Error Components Logit Model
3. Latent Class Multinomial Logit Model

The data used in this set of examples are experimental, and are carefully generated by an underlying latent class model in which the class probabilities depend on age and sex, and the choices depend on fashion, quality and price, exactly as specified below. The results are shown in Figure 67.

```
LCLOGIT ; Lhs = Choice ; Choices = Brand1,Brand2,Brand3,None
Rhs = asc4,fash,qual,price,pricesq ; Pds = 8
Lcm = male,age25,age39 ; Pts = 3$
```

| Coefficient | Standard Error | z | Prob (|z|>z) | 95% Confidence Interval |
|-------------|----------------|---|----------|------------------------|
| ASC1        | -1.4475***     | 2.52 | 2.32     | 0.022                 |
| FASH        | 0.0145***      | 0.02 | 2.32     | 0.022                 |
| QUAL1       | 0.2743         | 0.53 | 0.37     | 0.71                  |
| PRICE1      | 0.6621         | 0.31 | 2.13     | 0.033                 |

This is THETA(01) in class probability model

| Coefficient | Standard Error | z | Prob (|z|>z) | 95% Confidence Interval |
|-------------|----------------|---|----------|------------------------|
| ASC2        | -0.4323        | 0.43 | 0.50     | 0.61                  |
| FASH2       | 1.2206***      | 0.05 | 2.49     | 0.014                 |
| QUAL2       | 1.0276***      | 0.54 | 1.93     | 0.055                 |
| PRICE2      | -0.7736        | 0.41 | 1.88     | 0.060                 |

This is THETA(21) in class probability model

| Coefficient | Standard Error | z | Prob (|z|>z) | 95% Confidence Interval |
|-------------|----------------|---|----------|------------------------|
| ASC3        | 0.2743         | 0.53 | 0.37     | 0.71                  |
| FASH3       | 0.6621         | 0.31 | 2.13     | 0.033                 |
| QUAL3       | 0.2743         | 0.53 | 0.37     | 0.71                  |
| PRICE3      | -0.4323        | 0.43 | 0.50     | 0.61                  |

This is THETA(31) in class probability model

| Coefficient | Standard Error | z | Prob (|z|>z) | 95% Confidence Interval |
|-------------|----------------|---|----------|------------------------|
| ASC4        | -0.4323        | 0.43 | 0.50     | 0.61                  |
| FASH4       | 1.2206***      | 0.05 | 2.49     | 0.014                 |
| QUAL4       | 1.0276***      | 0.54 | 1.93     | 0.055                 |
| PRICE4      | -0.7736        | 0.41 | 1.88     | 0.060                 |

This is THETA(02) in class probability model

| Coefficient | Standard Error | z | Prob (|z|>z) | 95% Confidence Interval |
|-------------|----------------|---|----------|------------------------|
| ASC5        | 0.2743         | 0.53 | 0.37     | 0.71                  |
| FASH5       | 0.6621         | 0.31 | 2.13     | 0.033                 |
| QUAL5       | 0.2743         | 0.53 | 0.37     | 0.71                  |
| PRICE5      | -0.4323        | 0.43 | 0.50     | 0.61                  |

This is THETA(12) in class probability model

| Coefficient | Standard Error | z | Prob (|z|>z) | 95% Confidence Interval |
|-------------|----------------|---|----------|------------------------|
| ASC6        | -0.4323        | 0.43 | 0.50     | 0.61                  |
| FASH6       | 1.2206***      | 0.05 | 2.49     | 0.014                 |
| QUAL6       | 1.0276***      | 0.54 | 1.93     | 0.055                 |
| PRICE6      | -0.7736        | 0.41 | 1.88     | 0.060                 |

This is THETA(12) in class probability model

| Coefficient | Standard Error | z | Prob (|z|>z) | 95% Confidence Interval |
|-------------|----------------|---|----------|------------------------|
| ASC7        | 0.2743         | 0.53 | 0.37     | 0.71                  |
| FASH7       | 0.6621         | 0.31 | 2.13     | 0.033                 |
| QUAL7       | 0.2743         | 0.53 | 0.37     | 0.71                  |
| PRICE7      | -0.4323        | 0.43 | 0.50     | 0.61                  |

This is THETA(21) in class probability model

| Coefficient | Standard Error | z | Prob (|z|>z) | 95% Confidence Interval |
|-------------|----------------|---|----------|------------------------|
| ASC8        | -0.4323        | 0.43 | 0.50     | 0.61                  |
| FASH8       | 1.2206***      | 0.05 | 2.49     | 0.014                 |
| QUAL8       | 1.0276***      | 0.54 | 1.93     | 0.055                 |
| PRICE8      | -0.7736        | 0.41 | 1.88     | 0.060                 |

This is THETA(21) in class probability model

| Coefficient | Standard Error | z | Prob (|z|>z) | 95% Confidence Interval |
|-------------|----------------|---|----------|------------------------|
| ASC9        | 0.2743         | 0.53 | 0.37     | 0.71                  |
| FASH9       | 0.6621         | 0.31 | 2.13     | 0.033                 |
| QUAL9       | 0.2743         | 0.53 | 0.37     | 0.71                  |
| PRICE9      | -0.4323        | 0.43 | 0.50     | 0.61                  |

This is THETA(31) in class probability model

| Coefficient | Standard Error | z | Prob (|z|>z) | 95% Confidence Interval |
|-------------|----------------|---|----------|------------------------|
| ASC10       | -0.4323        | 0.43 | 0.50     | 0.61                  |
| FASH10      | 1.2206***      | 0.05 | 2.49     | 0.014                 |
| QUAL10      | 1.0276***      | 0.54 | 1.93     | 0.055                 |
| PRICE10     | -0.7736        | 0.41 | 1.88     | 0.060                 |

This is THETA(31) in class probability model

```

Figure 67. Estimated Latent Class Model

56
IX. Tools

*NLOGIT* provides a variety of tools that can be used with the model estimation commands or to create new estimators or statistics.

A. Scientific Calculator – The CALC Command

*NLOGIT*’s scientific calculator is an important tool. In the following application we use it to compute the F ratio for a Chow test, then look up the ‘p value’ for the test by computing a probability from the F distribution. Note that the named scalars computed with the CALC commands are added to the project, in the scalars list.

![Figure 68. Chow Test Using Calculator](image)

You can invoke the calculator with a CALC command that you put on your editing screen, such as CALC;1+1$, then highlight and submit with GO, as usual.

A Tip: CALC is a programming tool. As such, you will not always want to see the results of CALC. The CALC commands in the example above that pick up the sums of squares and the one that computes 1+1, do not display the result. If you want to see the result of CALC, add the word ;List to the command, as in CALC;List;1+1$ and in the commands above that compute the F statistic and the critical value.

The other way you can invoke the calculator is to use Tools→Scalar Calculator to open a calculator window. This would appear like the one below. When you use a calculator window, the results are always listed on the screen.
The example in Figure 69 computes two results, the sum of one and one and the rank of the covariance matrix for the coefficients in the most recently computed regression.

In addition to the full range of algebra, CALC provides approximately 100 functions, such as the familiar ones, log, exp, abs, sqr, and so on, plus functions for looking up table values from the normal, $t$, $F$, and chi-squared distributions, functions for computing integrals (probabilities) from these distributions, some matrix functions such as rank and trace, and many other functions.

Any result that you calculate with CALC can be given a name, and used later in any other context that uses numbers. Note, for example, in the example in Figure 68, the scalars that are the sums of squares are used in the later command that computes the $F$ statistic. All model commands, such as REGRESS, compute named results for the calculator. You can see the full list of these under the heading ‘Scalars’ in the project window shown in Figure 47. After you use REGRESS to compute a regression, these additional results are computed and saved for you to use later. Note, once again, the example in Figure 68. Each of the three REGRESS commands is followed by a CALC command that uses the quantity SUMSQDEV. In each case, this value will equal the sum of squared residuals from the previous regression. That is how we accumulate the three values that we need for the Chow test. Other statistics, $YBAR$, LOGL, and so on, are also replaced with the appropriate values when you use REGRESS or any other model command. The other model commands, such as PROBIT, also save some results, but in many cases, not all of them. For example, PROBIT does not save a sum of squared deviations, but it does save LOGL and KREG, which is the number of coefficients.

B. Matrix Algebra

The other major tool you will use is the matrix algebra calculator. NLOGIT provides a feature that will allow you to do the full range of matrix algebra computations. To see how this works, here is a fairly simple application: The LM statistic for testing the hypothesis that $\sigma^2 = h(z'\gamma)$ against the null hypothesis that $\sigma^2$ is constant in a classical regression model is computed as $LM = 0.5 \cdot g'(Z(Z'Z)^{-1}Z'g$ where $g$ is a vector of $n$ observations on $[e_i^2/(e'e/n) - 1]$ with $e_i$ the least squares residual in the regression of $y$ on $X$, and $Z$ is the set of variables in the variance function. A general set of instructions that could be used to compute this statistic are

```
NAMELIST ; x = the list of variables ; z = the list of variables $
REGRESS ; Quietly ; Lhs = y ; Rhs = x ; Res = e $
CREATE ; g = (e^2/(sumsqdev/n)-1) $
MATRIX ; list ; lm = .5 * g'z * <z'z> * z'g $
```

The NAMELIST command defines the matrices used. REGRESS (quietly) computes the residuals and calls them $e$. (There is a matrix command that will do this as well.) CREATE uses the regression results to compute the $n$ observations on $g$. Finally, MATRIX does the actual calculation. The MATRIX command works the same as CALC, either in the editor screen or in its own Tools window.

There are only a few things you need to get started using NLOGIT’s matrix algebra program. The first is how to define a data matrix, such as $X$ in the example above. The columns of a data matrix are variables, so, as you can see in the example, the NAMELIST command defines the columns of a data matrix. A single variable defines a data matrix with one column (i.e., a data vector) – note the use of the variable $g$ in the example.
• The rows of a data matrix are the observations in the current sample, whatever that happens to be at the time. That means that all data matrices change when you change the sample. For example, NAMELIST ; x=one,age,educ,income $ for our full healthcare data set defines a 2309×4 data matrix. When it is followed by SAMPLE;1-500$ , x becomes a 500×4 matrix.

• Data matrices can share columns. For example, with the x just defined, we might also have a NAMELIST;z=one,age,educ,income,married,hhkids $ Thus, x and z share four columns.

• In matrix algebra, the number 1 will represent a column of ones. Thus, if x is a variable, you could compute its mean with MATRIX;List;Meanx=1/n*x'1$. In defining a data matrix, as we did above, you may include ‘one’ to carry a column of ones.

There are many matrix operators. The major ones you need to know are

1. +, - , * for the usual addition, subtraction, and multiplication.
   The program will always check conformability. Note, row and column vectors are different.
2. ‘ (apostrophe) for transposition
3. <,> for inversion
4. [variable] for a diagonal matrix in a quadratic form.

The last of these allows you to compute a result that involves a possibly huge diagonal matrix. For example, in a Poisson regression context, the asymptotic covariance matrix of the MLE is

$$\text{Asy.Var}[\beta] = (X'AX)^{-1}$$

Where X is the n×K data matrix and A is a diagonal matrix with $\lambda_i = \exp(\beta'x_i)$ on the diagonal. If you have, say, 1,000,000 observations (you might), then A is a 1,000,000×1,000,000 matrix that save for the tiny percentage of values that are on the diagonal, is a matrix of zeros. Obviously, you do not want to create A in your computer’s memory. But, the syntax above allows you to do that. The matrix result is actually

$$\text{Asy.Var}[\beta] = [\Sigma \lambda_i x_i x_i']^{-1}$$

which is never larger than K×K. NLOGIT’s matrix syntax reveals this to the program. The matrix command would be

MATRIX ; AsyVarb = < x' [ lambdai ] x >$

You could compute this with millions of observations.

When you compute a moment matrix, such as $X'X$, you need not both transpose and multiply. This would involve having a copy of X that is the transpose of X. Again, this is a superfluous waste of space. The command $X'X$ means exactly what it looks like. The apostrophe is an operator that dictates how the result is to be computed.

In order to define a matrix with specific values in it, you use

$$\text{MATRIX ; NAME = [ row 1 / row 2 / ... ]}$

Within a row, values are separated by commas; rows are separated by slashes, and the whole thing is enclosed in square brackets. An example appears below. If the matrix is symmetric, you can define the matrix by its lower triangle – the first row has one element, the second has two elements, and so on.

In the same way that every model command creates some scalar results, every model command also creates at least two matrices, one named B which is the coefficient vector estimated, and one called VARB which is the estimated covariance matrix. You can use these in your matrix commands just like any other matrix. To compute the Poisson covariance matrix in the example immediately above, you could use
NAMELIST ; x = the list of variables $
POISSON ; Lhs = y ; Rhs = x ; Keep = lambdai $
MATRIX ; List ; AsyVarb = <x'|lambdai|x> ; varb $

The display would reveal that the matrix we computed, AsyVarb, and the internally computed matrix, varb, are identical.

For another example, here is a way to compute the restricted least squares estimator,

$$
b^* = b - (X'X)^{-1}R'[R(X'X)^{-1}R]^{-1}(Rb - q).$$

For a specific example, suppose we regress y on a constant, $x_1, x_2$, and $x_3$, then compute the coefficient vector subject to the restrictions that $b_2 + b_3 = 1$ and $b_1 = 0$. We will also compute the Wald statistic for testing this restriction,

$$W = (Rb-q)'[R s^2(X'X)^{-1}R]^{-1}(Rb-q).$$

Note that both examples use a shortcut for a quadratic form in an inverse.

NAMELIST ; x = one, x1, x2, x3 $
REGRESS ; Lhs = y ; Rhs = X $
MATRIX ; r = [0,1,1,0 / 0,0,0,1] ; q = [1/0] $
MATRIX ; m = r*b - q ; d = r*<x'x>*r' $
; br = b - <x'x>*r'*<d>m $
; w = m'*<d>m $

In addition to the operators and standard features of matrix algebra, there are numerous functions that you might find useful. These include ROOT(symmetric matrix), CXRT(any matrix) for complex roots, DTRM(matrix) for determinant, SQRT(matrix) for square root and over 100 others.

C. Procedures

A procedure is a group of commands that you can collect and give a name to. To execute the commands in the procedure, you simply use an EXECUTE command. To define a procedure, just place the group of commands in your editor window between PROCEDURE$ and ENDPROCEDURE$ commands, then run the whole group of them. They will not be carried out at that point; they are just stored and left ready for you to use later. For example, the application above that computes a restricted regression and reports the results could be made into a procedure as follows:

PROCEDURE $ 
REGRESS ; Lhs = y ; Rhs = X $
MATRIX ; r = [0,1,1,0 / 0,0,0,1] ; q = [1/0] $
MATRIX ; m = r*b - q ; d = r*<x'x>*r' $
; br = b - <x'x>*r'*<d>m $
; w = m'*<d>m $
ENDPROCEDURE $ 

Now, to compute the estimator, we would define X, y, r, and q, then use the EXECUTE command;

NAMELIST ; X = the set of variables $ 
CREATE ; y = the dependent variable $ 
MATRIX ; r = the matrix of constraints $ 
; q = the vector on the RHS of the constraints $ 
EXECUTE $ 

To use a different model, we’d just redefine X, y, R, and q, then execute again.
Since the commands for the procedure are just sitting on the screen waiting for us to Run them with a couple of mouse clicks, this really has not gained us very much. There are several better reasons for using procedures. The EXECUTE command can be made to request more than one run of the procedure, procedures can be written with ‘adjustable parameter lists,’ so that you can make them very general, and can change the procedure very easily. Repetitions of procedures can be used to develop bootstrap estimators of sample statistics.

The following computes a Chow test of structural change based on an X matrix, a y variable, and a dummy variable, d, which separates the sample into two subsets of interest. We’ll write this as a ‘subroutine’ with adjustable parameters. Note that this routine does not actually report the results of the three least squares regressions. To add this to the routine, the CALC commands which obtain sums of squares could be replaced with REGRESS ;Lhs = y ; Rhs = X $ then CALC ; ee = sumsqdev $ In this application, we have used a feature of PROC that allows it to accept adjustable parameters.

```f90
/* Procedure to carry out a Chow test of structural change.
   Inputs: X = namelist that contains full set of independent variables
           y = dependent variable
           d = dummy variable used to partition the sample
   Outputs F = sample F statistic for the Chow test
*/
F95 = 95th percentile from the appropriate F table.
PROC = ChowTest(X,y,d) $
CALC ; k = Col(X) ; Nfull = N $
INCLUDE ; New ; D = 1 $
CALC ; ee1 = Ess(X,y) $
INCLUDE ; New ; D = 0 $
CALC ; ee0 = Ess(X,y) $
SAMPLE ; All $  
CALC ; ee = Ess(X,y) $
CALC ; List $; F = ((ee-(ee1+ee0))/K) / (ee/(Nfull-2*K) ) ; F95 = Ftb(.95,K, (Nfull-2*K)) $
ENDPROC$
```

Now, suppose we wished to carry out the test of whether the labor supply behaviors of men and women are the same. The commands might appear as follows:

NAMELIST ; HoursEqn = One,Age,Exper,Kids $ EXECUTE ; Proc = ChowTest(HoursEqn,Hours,Sex) $

A Tip: The preceding illustrates a particular calculation using a procedure. The Chow test (or its maximum likelihood equivalent for nonlinear models) can be carried out with a single command, such as

REGRESS ; For[(test) female = *,0,1] ; Lhs = y ; Rhs = x $

One of the main uses of procedures is to carry out repetitions of instructions. The following example illustrates. The next section extends this idea to bootstrapping estimators. The procedure in the example is applied in Figure 70.
/* The data set consists of G groups. We wish to estimate a logit model of y on X for each group and arrange the coefficient vectors in the rows of a matrix named BG. There is a variable named GROUP that indexes the groups. We do not know G. That is to be determined. */

NAMELIST ; x = the group of variables $ 
CREATE ; y = the variable $
CALC ; g = max(group) ; k = col(x) $ Learn g and k from the model setup.
MATRIX ; bg = init(g,k,0.0) $ Matrix where where we will stack the coefficients
PROCEDURE $
LOGIT ; If[group = i] ; Quietly ; Lhs = y ; Rhs = x$
MATRIX ; bg(i,*') = b' $ Puts i'th coefficient vector in i'th row of matrix.
ENDPROC $
EXECUTE ; i = 1,g $ Executes for i = 1,2,...,g.

In the example below, ‘group’ is a random discrete uniform(1,10) variable, i.e., CREATE; group = rnd(10) $

Figure 70. Repeated Execution of a Procedure

D. Bootstrapping

You can use procedures to compute bootstrap results for any scalar or vector that you compute using data. This can be a coefficient vector, a test statistic, or any other result that is computed using a sample of data. The general form of the procedure is as follows:

... any preliminary setup
PROCEDURE $
... compute the scalar with CALC or the vector with MATRIX.
... This part of the procedure may contain as many commands and
... calculations as needed. It needs only to produce the result to be
... examine with a name, to be used later.
ENDPROC $ 
EXEC ; n = number of bootstrap replications ; Bootstrap = the name $
The procedure is actually executed \(n+1\) times, first with the full original sample, then \(n\) times with the bootstrap samples. In the following example, we compute the vector of partial effects in a Poisson regression and bootstrap a covariance matrix. (Partial effects for a Poisson regression is a built-in procedure in {\proglang{NLOGIT}} – we do this here just to illustrate the method.)

\begin{verbatim}
NAMELIST ; x = age, educ, income, hlthsat $
PROCEDURE $
POISSON ; quietly ; Lhs = docvis ; Rhs = x, one ; keep = lambdai $
CALC ; apescale = xbr(lambdai) $
MATRIX ; ape = apescale ^ b(1:4) $
ENDPROC $
EXEC ; n = 50 ; bootstrap = ape $
\end{verbatim}

Figure 71. Results of Bootstrap Iterations

When you compute bootstrap replicates such as those shown in Figure 71, {\proglang{NLOGIT}} also creates a matrix named \texttt{BOOTSTRP} that contains the actual replicates. Figure 72 shows part of the results for the experiment in Figure 71.
E. Displaying Results

*NLOGIT* provides several ways to display estimation results (and several formats, including export to *Excel* and formatted tables that can be exported to editors such as *Word*). To produce a standard output table for a set of estimates and the estimated covariance matrix, you need the estimates, the matrix, labels for the estimates (optional) and, perhaps, a title. Figure 73 shows how to construct a DISPLAY command for our bootstrap results in Figure 71. The command is

```
DISPLAY ; parameters = ape [the name of the coefficient vector]  
; covariance = varb [the name of the covariance matrix]  
; labels = x [here, x provides a set of names, not the actual data]  
; title = Bootstrap … $ [the desired title]
```

Figure 73. Display of Estimation Results
E. WALD, SIMULATE and Standard Errors for Nonlinear Functions

Two devices, WALD and SIMULATE, are provided for computing functions of parameters and standard errors for nonlinear functions. Both of them compute linear or nonlinear functions and standard errors usually using the delta method. (The method of Krinsky and Robb is also available.) Functions can be any desired computation using a parameter vector and the data.

1. The WALD Command

WALD is used for computing multiple functions and can be used to test hypotheses about functions of parameters. To illustrate, we manipulate the average partial effects shown in Figures 71 and 73. The WALD command to examine what is actually not a useful function would appear thusly:

```
WALD ; parameters = ape
; covariance = varb
; labels = ca,ce,ci,ch
; fn1=ca*exp(ca'x) + phi(ca) $
```

A Tip: In the function definition above, x is a namelist with 4 names that was defined above in part D, x=age,educ,income,hlthsat. The parameter vector is (ca,ce,ci,ch). The construction ca’x uses the parameters beginning with ca and x beginning with the first variable to compute the inner product. When one of the two components is shorter than the other, the shorter list is used. Thus, ce’x = ce*age+ci*educ+ch*income. If we defined z=age,educ, then ca’z would equal ca*age+ce*educ.

WALD requires the parameter vector, covariance matrix, labels, and up to 50 function definitions. As seen in the top panel of Figure 74, WALD computes the function at the means of the data using the current sample, and uses the delta method to compute standard errors and confidence intervals. By adding ;Average to the command, you can request that the average function value be computed, rather than the functions at the averages. This appears in the lower panel of Figure 74. WALD also computes the chi squared test of the null hypothesis that all of the functions are jointly zero. Note, in Figure 74, there is one function – the Wald statistic in this case is the square of the z statistic.

```
<table>
<thead>
<tr>
<th>WaldFuns</th>
<th>Coefficient</th>
<th>Standard Error</th>
<th>Prob</th>
<th>95% Confidence Interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fcn(1)</td>
<td>51703***</td>
<td>00696</td>
<td>74.51</td>
<td>.0000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.50421</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.5345</td>
</tr>
</tbody>
</table>

Note: *** *** * * * ** Significance at 1%, 5%, 10% level
```

```
<table>
<thead>
<tr>
<th>WaldFuns</th>
<th>Coefficient</th>
<th>Standard Error</th>
<th>Prob</th>
<th>95% Confidence Interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fcn(1)</td>
<td>51703***</td>
<td>00696</td>
<td>74.51</td>
<td>.0000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.50421</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.5345</td>
</tr>
</tbody>
</table>

Note: *** *** * * * ** Significance at 1%, 5%, 10% level
```

Figure 74. WALD Command for Analyzing Nonlinear Functions
2. The SIMULATE Command

The SIMULATE command shown in Section VII.B.2 can also be used to analyze functions of parameter estimates. The base cases are given the same result as WALD, as shown in Figure 75 for this example – note the function analyzed is the same as used in WALD.

```
SIMULATE ; parameters = ape ; covariance = varb
   ; labels = ca,ce,ci,ch
   ;function=ca*exp(ca'x) + phi(ca)
```

```
Figure 75. Analyzing a Function with SIMULATE
```

SIMULATE computes the average function as opposed to WALD which computes the function at the means. As noted, WALD will compute the average function if the command contains ;Average. SIMULATE will compute the function at the means if the command contains ;Means.

3. WALD or SIMULATE - Which Should You Use?

For computing a function and appropriate standard errors, WALD and SIMULATE give the same answers. They differ as follows:

- WALD can be used to compute the chi squared test statistic for testing the hypothesis that the functions are all zero (simultaneously)
- WALD can analyze up to 50 functions in the single command.
- SIMULATE has many options for analyzing scenarios and simulating a function over a variety of different settings of the variables in the equation.
- SIMULATE can plot function values as well as listing them.

An example of a more elaborate use of SIMULATE appears in Figure 76. The command is as follows:

```
SIMULATE ; parameters = ape ; covariance = varb
   ; labels = ca,ce,ci,ch
   ;function=ca*exp(ca'x) + phi(ca)
   ;scenario: & educ=12(1)20
   ;plot
```

Figure 76. Analyzing a Scenario with SIMULATE