11. Introduction to Stata

Data Sets Used for Stata Exercises

This course works extensively with Stata, using a subset of information from the Bangladesh Household Survey 1991/92–1998/99, conducted jointly by the Bangladesh Institute of Development Studies and the World Bank. The information was collected at individual, household, and community levels. What follows is a description of the data sets and file structure used for these exercises. The data files for these exercises can be downloaded from the World Bank Web site, and as mentioned earlier represent subsets of the actual data for purposes of these exercises only. The Web site is accessible by the following steps:

1. Go to http://econ.worldbank.org
2. In the lower right hand corner, under *Resources* click on: *People and Bios*
3. Click on: *Research Staff (alphabetical list)*
4. Under "K" select: *Shahidur R. Khandker*
5. Click on the link for the book: *Handbook on Impact Evaluation.*

Alternatively, the Web site is accessible at: http://go.worldbank.org/FE8098BI60.

This location has the original full dataset and the subset files which pertain to the exercises.

File Structure

These exercises use and generate many files. There are mainly three types of Stata files. Some contain data sets (identified by the suffix .dta), others contain Stata programs (identified by the suffix .do), and yet others contain a record and output of the work done in Stata (identified by the suffix .log). To keep these files organized, the following directory structure has been created:

```
c:\eval
c:\eval\data
c:\eval\do
c:\eval\log
```

File Descriptions

The data files are located under c:\eval\data. There are three data files:

1. *hh_91.dta.* This file comprises the 1991 household data that contain 826 households (observations). These data have 24 variables on information at the household

(head's education, land ownership, expenditure, and so on) and village (infrastructure, price information of the main consumer goods, and so on) levels.
2. *hh_98.dta.* This file is the 1998 panel version of hh_91.dta. It includes 303 new households, making the total number of households (observations) 1,129. These data contain the same household- and village-level variables as hh_91.dta.
3. *hh_9198.dta.* This is a panel data set restricted to the 826 households interviewed in both years. It is in a time-series format.

A list of variables that are included in the data set appears in figure 11.1.

The .do folder has the program (.do) files specific to different impact evaluation techniques. These files contain all Stata code needed to implement the examples of the corresponding chapter (Microsoft Word file) that walks through the hands-on exercises. A segment of the .do file can be run for a particular example or case, or the whole .do file can be run to execute all examples in the chapters.

The .log folder contains all outputs generated by running the .do files.

Beginning Exercise: Introduction to Stata

Stata is a statistical software package that offers a large number of statistical and econometric estimation procedures. With Stata, one can easily manage data and apply standard statistical and econometric methods such as regression analysis and limited dependent variable analysis to cross-sectional or longitudinal data.

Getting Started

Start a Stata session by double-clicking on the Stata icon on your desktop. The Stata computing environment comprises four main windows. The size and shape of these windows may be changed, and they may be moved around on the screen. Figure 11.2 shows their general look and description.

In addition to these windows, the Stata environment has a menu and a toolbar at the top (to perform Stata operations) and a directory status bar at the bottom (that shows the current directory). You can use the menu and the toolbar to issue different Stata commands (such as opening and saving data files), although most of the time using the Stata Command window to perform those tasks is more convenient. If you are creating a log file (discussed in more detail later), the contents can be displayed on the screen, which is sometimes useful if you want to go back and see earlier results from the current session.

Opening a Data Set

You can open a Stata data set by entering following command in the Stata Command window:

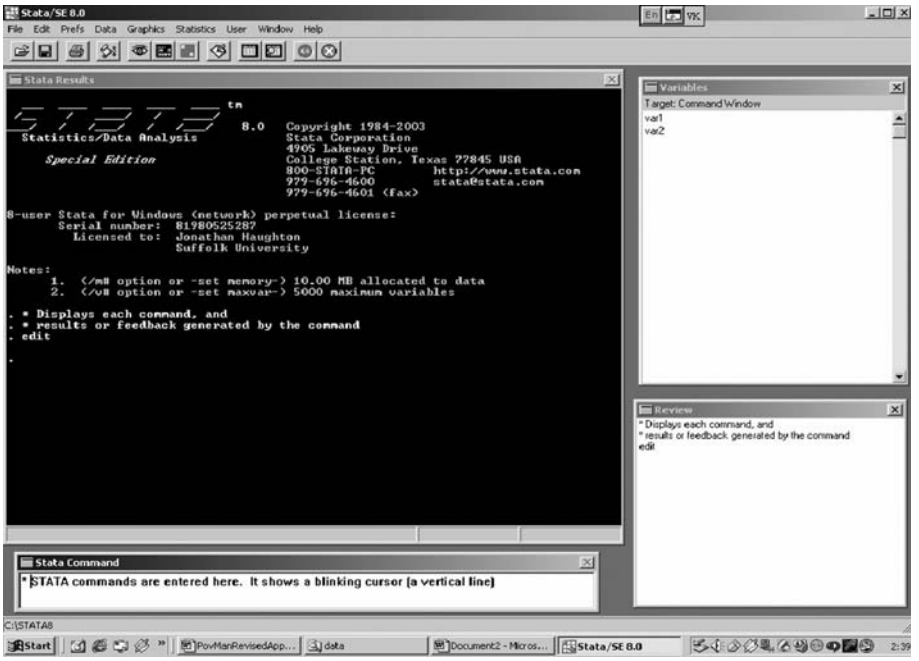
```
use c:\eval\data\hh_98.dta
```

Figure 11.1 Variables in the 1998/99 Data Set

```
Contains data from hh_98.dta
  obs:         1,129
 vars:            24                          1 Apr 2009 12:04
 size:       119,674 (99.9% of memory free)
-------------------------------------------------------------------------------
              storage  display     value
variable name   type   format      label      variable label
-------------------------------------------------------------------------------
nh              double %7.0f                  HH ID
year            float  %9.0g                  Year of observation
villid          double %9.0g                  Village ID
thanaid         double %9.0g                  Thana ID
agehead         float  %3.0f                  Age of HH head: years
sexhead         float  %2.0f                  Gender of HH head: 1=M, 0=F
educhead        float  %2.0f                  Education of HH head: years
famsize         float  %9.2f                  HH size
hhland          float  %9.0g                  HH land: decimals
hhasset         float  %9.0g                  HH total asset: Tk.
expfd           float  %9.0g                  HH per capita food expenditure:
                                                Tk/year
expnfd          float  %9.0g                  HH per capita nonfood
                                                expenditure: Tk/year
exptot          float  %9.0g                  HH per capita total
                                                expenditure: Tk/year
dmmfd           byte   %8.0g                  HH has male microcredit
                                                participant: 1=Y, 0=N
dfmfd           byte   %8.0g                  HH has female microcredit
                                                participant: 1=Y, 0=N
weight          float  %9.0g                  HH sampling weight
vaccess         float  %9.0g                  Village is accessible by road
                                                all year: 1=Y, 0=N
pcirr           float  %9.0g                  Proportion of village land
                                                irrigated
rice            float  %9.3f                  Village price of rice: Tk./kg
wheat           float  %9.3f                  Village price of wheat: Tk./kg
milk            float  %9.3f                  Village price of milk: Tk./liter
potato          float  %9.3f                  Village price of potato: Tk./kg
egg             float  %9.3f                  Village price of egg: Tk./4
                                                counts
oil             float  %9.3f                  Village price of edible oil:
                                                Tk./kg
-------------------------------------------------------------------------------
Sorted by:  nh
```

Source: Bangladesh Institute of Development Studies–World Bank Household Survey 1998/99.

Figure 11.2 The Stata Computing Environment



Source: Screenshot of Stata window.

You can also click on File and then Open and then browse to find the file you need. Stata responds by displaying the following in the Stata Results window:

```
. c:\eval\data\hh_98.dta
```

The first line repeats the command you enter, and absence of an error message in a second line implies the command has been executed successfully. From now on, only the Stata Results window will be shown to demonstrate Stata commands. The following points should be noted:

- Stata assumes the file is in Stata format with an extension .dta. Thus, typing "hh_98" is the same as typing "hh_98.dta."
- Only one data set can be open at a time in Stata. In this case, for example, if another data set hh_91.dta is opened, it will replace hh_98.dta with hh_91.dta.
- The preceding command assumes that the file hh_98.dta is not in the current directory. To make c:\eval\data the current directory and then open the file as before, enter the following commands:

```
. cd c:\eval\data
. use hh_98
```

If the memory allocated to Stata (which is by default 1,000 kilobytes, or 1 megabyte) is too little for the data file to be opened, as is typically the case when with large household survey data sets, an error message such as the following will appear:

```
. use hh_98
no room to add more observations
r(901);
```

The third line displays the code associated with the error message. All error messages in Stata have associated codes like this one; further explanations are available in the Stata reference manuals. In this case, more memory must be allocated to Stata. The following commands allocate 30 megabytes to Stata and then try again to open the file:

```
. set memory 30m
[This generates a table with memory information]
. use hh_98
```

Because the file opens successfully, allocated memory is sufficient. If you continue to get an error message, you can use a larger amount of memory, although it may slow down your computer somewhat. Note that the "set memory" command works only if no data set is open (in memory). Otherwise, you will get following error message:

```
. use hh_98
. set memory 10m
no; data in memory would be lost
r(4);
```

You can clear the memory by using one of the two commands: "clear" or "drop _all." The following demonstration shows the first command:

```
. use hh_98
. set memory 10m
no; data in memory would be lost
r(4);
. clear
. set memory 10m
```

Saving a Data Set

If you make changes in an open Stata data file and want to save those changes, you can do so by using the Stata "save" command. For example, the following command saves the hh_98.dta file:

```
. save hh_98, replace
file hh_98.dta saved
```

You can optionally omit the file name here (just "save, replace" is good enough). If you do not use the replace option, Stata does not save the data but issues the following error message:

```
. save hh_98
file hh_98.dta already exists
r(602);
```

The replace option unambiguously tells Stata to overwrite the preexisting original version with the new version. If you do *not* want to lose the original version, you have to specify a different file name in the "save" command.

Exiting Stata

An easy way to exit Stata is to issue the command "exit." However, if you have an unsaved data set open, Stata will issue the following error message:

```
. exit
no; data in memory would be lost
r(4)
```

To remedy this problem, you can save the data file and then issue the "exit" command. If you really want to exit Stata without saving the data file, you can first clear the memory (using the "clear" or "drop _all" command as shown before) and issue the "exit" command. You can also simplify the process by combining two commands:

```
. exit, clear
```

Stata Help

Stata comes with an excellent multivolume set of manuals. However, the on-computer help facility in Stata is extensive and very useful; if you have access to the Web, an even larger set of macros and other useful information are available.

From within Stata, if you know which command or keyword you want the help information about, you can issue the command "help" followed by the command name or keyword. This command works only if you type the full command name or keyword with no abbreviations. For example, the following command will not work:

```
. help mem
help for mem not found
try help contents or search mem
```

However, this command will:

```
. help memory
[output omitted]
```

If you cannot recall the full command name or keyword, or if you are not sure about which command you want, you can use the command "lookup" or "search" followed by the command name or keyword. So the following will work:

```
. search mem
[output omitted]
```

This command will list all commands associated with this keyword and display a brief description of each of those commands. Then you can pick the command that you think is relevant and use help to obtain the specific reference.

The Stata Web site (http://www.stata.com) has excellent help facilities, such as an online tutorial and frequently asked questions (FAQ).

Notes on Stata Commands

Here are some general comments about Stata commands:

- Stata commands are typed in lowercase.
- All names, including commands or variable names, can be abbreviated as long as no ambiguity exists. For example, "describe," "des," and simply "d" do the same job because no confusion exists.
- In addition to typing, some keystrokes can be used to represent a few Stata commands or sequences. The most important of them are the Page-Up and Page-Down keys. To display the previous command in the Stata Command window, you can press the Page-Up key. You can keep doing so until the first command of the session appears. Similarly, the Page-Down key displays the command that follows the currently displayed command in the Stata Command window.
- Clicking once on a command in the Review window will put it into the Stata Command window; double-clicking it will tell Stata to execute the command. This can be useful when commands need to be repeated or edited slightly in the Stata Command window.

Working with Data Files: Looking at the Content

To go through this exercise, open the hh_98.dta file; examples from this data file are used extensively.

Listing the Variables

To see all variables in the data set, use the "describe" command (in full or abbreviated):

```
. describe
```

This command provides information about the data set (name, size, number of observations) and lists all variables (name, storage format, display format, label).

To see just one variable or list of variables, use the describe command followed by the variable name or names:

```
. desc nh villid
                  storage   display    value
variable name      type      format    label      variable label
------------------------------------------------------------------
nh                 double    %7.0f                 HH ID
villid             double    %9.0g                 Village ID
```

As you can see, the describe command shows also the variable type and length, as well as a short description of the variable (if available). The following points should be noted:

- You can abbreviate a list of variables by typing only the first and last variable names, separated by a hyphen (-); the Variables window shows the order in which the variables are stored. For example, to see all variables from "nh" to "famsize," you could type

  ```
  . describe nh-famsize
  ```

- The wild card symbol (*) is helpful to save some typing. For example, to see all variables that start with "exp," you could type

  ```
  . describe exp*
  ```

- You can abbreviate a variable or variable list this way in any Stata command (where it makes sense), not just in the "describe" command.

Listing Data

To see actual data stored in the variables, use the "list" command (abbreviated as "l"). If you type the command "list" by itself, Stata will display values for all variables and all observations, which may not be desirable for any practical purpose (and you may need to use the Ctrl-Break combination to stop data from scrolling endlessly across the screen). Usually you want to see the data for certain variables and for certain observations. This is achieved by typing a "list" command with a variable list and with conditions.

The following command lists all variables of the first three observations:

```
. list in 1/3
```

Here Stata displays all observations starting with observation 1 and ending with observation 3. Stata can also display data as a spreadsheet. To do so, use the two icons in the toolbar called Data Editor and Data Browser (fourth and third from right). Clicking one will cause a new window to pop up where the data will be displayed as a table, with observations as rows and variables as columns. Data Browser will only display the data, whereas you can edit data with Data Editor. The commands "edit" and "browse" will also open the spreadsheet window.

The following command lists household size and head's education for households headed by a female who is younger than 45:

```
. list famsize educhead if (sexhead==0 & agehead<45)
```

The prior statement uses two relational operators (== and <) and one logical operator (&). Relational operators impose a condition on one variable, while logical operators combine two or more relational operators. Table 11.1 shows the relational and logical operators used in Stata.

You can use relational and logical operators in any Stata command (where it makes sense), not just in the "list" command.

Summarizing Data

The very useful command "summarize" (which may be abbreviated "sum") calculates and displays a few summary statistics, including means and standard deviations. If no variable is specified, summary statistics are calculated for all variables in the data set. The following command summarizes the household size and education of the household head:

```
. sum famsize educhead
```

Stata excludes any observation that has a missing value for the variables being summarized from this calculation (missing values are discussed later). If you want to know the median and percentiles of a variable, add the "detail" option (abbreviated "d"):

```
. sum famsize educhead, d
```

A great strength of Stata is that it allows the use of weights. The weight option is useful if the sampling probability of one observation is different from that of another. In most household surveys, the sampling frame is stratified, where the first primary sampling units (often villages) are sampled, and conditional on the selection of primary sampling unit, secondary sampling units (often households) are drawn. Household surveys generally provide weights to correct for sampling design differences and sometimes data collection problems. The implementation in Stata is straightforward:

```
. sum famsize educhead [aw=weight]
```

Table 11.1 Relational and Logical Operators Used in Stata

Relational operators	Logical operators
> (greater than)	~ (not)
< (less than)	| (or)
== (equal)	& (and)
>= (greater than or equal)	
>= (less than or equal)	
!= or ~= (not equal)	

Source: Authors' compilation.

Here, the variable "weight" has the information on the weight to be given to each observation and "aw" is a Stata option to incorporate the weight into the calculation. The use of weights is discussed further in later chapter exercises.

For variables that are strings, the command "summarize" will not be able to give any descriptive statistics except that the number of observations is zero. Also, for variables that are categorical (for example, illiterate = 1, primary education = 2, higher education = 3), interpreting the output of the "summarize" command can be difficult. In both cases, a full tabulation may be more meaningful, which is discussed next.

Often, one wants to see summary statistics by groups of certain variables, not just for the whole data set. Suppose you want to see mean family size and education of household head for participants and nonparticipants. First, sort the data by the group variable (in this case, dfmfd). You can check this sort by issuing the "describe" command after opening each file. The "describe" command, after listing all the variables, indicates whether the data set is sorted by any variables. If no sorting information is listed or the data set is sorted by a variable that is different from the one you want, you can use the "sort" command and then save the data set in this form. The following commands sort the data set by the variable "dfmfd" and show summary statistics of family size and education of household head for participants and nonparticipants:

```
. sort dfmfd
. by dfmfd: sum famsize educhead [aw=weight]
```

A useful alternative to the "summary" command is the "tabstat" command, which allows you to specify the list of statistics you want to display in a single table. It can be conditioned by another variable. The following command shows the mean and standard deviation of the family size and education of household head by the variable "dfmfd":

```
. tabstat famsize educhead, statistics(mean sd) by(dfmfd)
```

Frequency Distributions (Tabulations)

Frequency distributions and cross-tabulations are often needed. The "tabulate" (abbreviated "tab") command is used to do this:

```
. tab dfmfd
```

The following command gives the gender distribution of household heads of participants:

```
. tab sexhead if dfmfd==1
```

In passing, note the use of the == sign here. It indicates that if the regional variable is identically equal to one, then do the tabulation.

The "tabulate" command can also be used to show a two-way distribution. For example, one might want to check whether any gender bias exists in the education of household heads. The following command is used:

```
. tab educhead sexhead
```

To see percentages by row or columns, add options to the "tabulate" command:

```
. tab dfmfd sexhead, col row
```

Distributions of Descriptive Statistics (Table Command)

Another very convenient command is "table," which combines features of the "sum" and "tab" commands. In addition, it displays the results in a more presentable form. The following "table" command shows the mean of family size and of education of household head, by their participation in microfinance programs:

```
. table dfmfd, c(mean famsize mean educhead)
----------------------------------------
HH has    |
female    |
microcred |
it        |
participa |
nt: 1=Y,  |
0=N       | mean(famsize)     mean(educhead)
----------+-----------------------------
        0 |      5.41                3
        1 |      5.21                2
----------------------------------------
```

The results are as expected. But why is the mean of "educhead" displayed as an integer and not a fraction? This occurs because the "educhead" variable is stored as an integer number, and Stata simply truncated numbers after the decimal. Look at the description of this variable:

```
. d educhead
              storage   display   value
variable name   type    format    label         variable label
------------------------------------------------------------------------
educhead        float     %2.0f    Education (years) of HH Head
```

Note that "educhead" is a float variable: its format (%2.0f) shows that its digits occupy two places, and it has no digit after the decimal. You can force Stata to reformat the display. Suppose you want it to display two places after the decimal, for a

three-digit display. The following command shows that command and the subsequent "table" command:

```
. format educhead %3.2f
. table dfmfd, c(mean famsize mean educhead)
-----------------------------------------
HH has    |
female    |
microcred |
it        |
participa |
nt: 1=Y,  |
0=N       | mean(famsize)     mean(educhead)
----------+------------------------------
        0 |     5.41              2.95
        1 |     5.21              1.75
-----------------------------------------
```

This display is much better. Formatting changes only the display of the variable, not the internal representation of the variable in the memory. The "table" command can display up to five statistics and variables other than the mean (such as the sum or minimum or maximum). Two-way, three-way, or even higher-dimensional tables can be displayed.

Here is an example of a two-way table that breaks down the education of the household head not just by region but also by sex of household head:

```
. table dfmfd sexhead, c(mean famsize mean educhead)
----------------------
HH has    |
female    |
microcred |
it        |  Gender of
participa |   HH head:
nt: 1=Y,  |   1=M, 0=F
0=N       |     0    1
----------+-----------
        0 |  4.09 5.53
          |  1.18 3.11
          |
        1 |  4.25 5.31
          |  0.59 1.88
----------------------
```

Missing Values in Stata

In Stata, a missing value is represented by a dot (.). A missing value is considered larger than any number. The "summarize" command ignores the observations with

missing values, and the "tabulate" command does the same, unless forced to include missing values.

Counting Observations

The "count" command is used to count the number of observations in the data set:

```
. count
 1129
.
```

The "count" command can be used with conditions. The following command gives the number of households whose head is older than 50:

```
. count if agehead>50
  354
.
```

Using Weights

In most household surveys, observations are selected through a random process and may have different probabilities of selection. Hence, one must use weights that are equal to the inverse of the probability of being sampled. A weight of *wj* for the *j*th observation means, roughly speaking, that the *j*th observation represents *wj* elements in the population from which the sample was drawn. Omitting sampling weights in the analysis usually gives biased estimates, which may be far from the true values.

Various postsampling adjustments to the weights are usually necessary. The household sampling weight that is provided in the hh.dta is the right weight to use when summarizing data that relates to households.

Stata has four types of weights:

- Frequency weights ("fweight"), which indicate how many observations in the population are represented by each observation in the sample, must take integer values.
- Analytic weights ("aweight") are especially appropriate when working with data that contain averages (for example, average income per capita in a household). The weighting variable is proportional to the number of persons over which the average was computed (for example, number of members of a household). Technically, analytic weights are in inverse proportion to the variance of an observation (that is, a higher weight means that the observation was based on more information and so is more reliable in the sense of having less variance).
- Sampling weights ("pweight") are the inverse of the probability of selection because of sample design.
- Importance weights ("iweight") indicate the relative importance of the observation.

The most commonly used are "pweight" and "aweight." Further information on weights may be obtained by typing "help weight."

The following commands show application of weights:

```
. tabstat famsize [aweight=weight], statistics(mean sd) by(dfmfd)
. table dfmfd [aweight=weight], contents(mean famsize sd famsize)
```

	Full sample		Participants		Nonparticipants	
	Mean	Standard deviation	Mean	Standard deviation	Mean	Standard deviation
Household size						
Per capita expenditure						
Per capita food expenditure						
Per capita nonfood expenditure						

Are the weighted averages very different from the unweighted ones?

Changing Data Sets

So far, discussion has been limited to Stata commands that display information in the data in different ways without changing the data. In reality, Stata sessions most often involve making changes in the data (for example, creating new variables or changing values of existing variables). The following exercises demonstrate how those changes can be incorporated in Stata.

Generating New Variables

In Stata the command "generate" (abbreviated "gen") creates new variables, while the command "replace" changes the values of an existing variable. The following commands create a new variable called "oldhead" and then set its value to one if the household head is older than 50 years and to zero otherwise:

```
. gen oldhead=1 if agehead>50
(775 missing values generated)
. replace oldhead=0 if agehead<=50
(775 real changes made)
```

What happens here is that, for each observation, the "gen" command checks the condition (whether household head is older than 50) and sets the value of the variable "oldhead" to one for that observation if the condition is true and to missing value otherwise. The "replace" command works in a similar fashion. After the "generate" command, Stata indicates that 775 observations failed to meet the condition, and after the "replace" command Stata indicates that those 775 observations have new values (zero in this case). The following points are worth noting:

- If a "gen" or "replace" command is issued without any conditions, that command applies to all observations in the data file.
- While using the generate command, one should take care to handle missing values properly.
- The right-hand side of the = sign in the "gen" or "replace" commands can be any expression involving variable names, not just a value. Thus, for instance, the command "gen young = (agehead<=32)" would create a variable called "young" that would take on the value of one if the head is 32 years of age or younger (that is, if the bracketed expression is true) and a value of zero otherwise.
- The "replace" command can be used to change the values of any existing variable, independently of the "generate" command.

An extension of the "generate" command is "egen." Like the "gen" command, the "egen" command can create variables to store descriptive statistics, such as the mean, sum, maximum, and minimum. The more powerful feature of the "egen" command is its ability to create statistics involving multiple observations. For example, the following command creates a variable "avgage" containing the average age of household heads for the whole data:

```
. egen avgage=mean(agehead)
```

All observations in the data set get the same value for "avgage." The following command creates the same statistics, but this time for male- and female-headed households separately:

```
. egen avgagemf=mean(agehead), by(sexhead)
```

Labeling Variables

You can attach labels to variables to give them a description. For example, the variable "oldhead" does not have any label now. You can attach a label to this variable by typing

```
. label variable oldhead "HH Head is over 50: 1=Y, 0=N"
```

In the "label" command, variable can be shortened to "var." Now to see the new label, type the following:

```
. des oldhead
```

Labeling Data

Other types of labels can be created. To attach a label to the entire data set, which appears at the top of the "describe" list, try

```
. label data "Bangladesh HH Survey 1998"
```

To see this label, type

```
. des
```

Labeling Values of Variables

Variables that are categorical, like those in "sexhead (1 = male, 0 = female)," can have labels that help one remember what the categories are. For example, using hh_98.dta, tabulating the variable "sexhead" shows only zero and one values:

```
. tab sexhead
```

```
 Gender of  |
  HH head:  |
  1=M, 0=F  |    Freq.   Percent    Cum.
------------+-----------------------------
         0  |     104     9.21      9.21
         1  |   1,025    90.79    100.00
------------+-----------------------------
      Total |   1,129   100.00
```

To attach labels to the values of a variable, two things must be done. First, define a value label. Then assign this label to the variable. Using the new categories for sexhead, type

```
. label define sexlabel 0 "Female" 1 "Male"
. label values sexhead sexlabel
```

Now, to see the labels, type

```
. tab sexhead
```

If you want to see the actual values of the variable "sexhead," which are still zeros and ones, you can add an option to not display the labels assigned to the values of the variable. For instance, try

```
. tab sexhead, nolabel
```

Keeping and Dropping Variables and Observations

Variables and observations of a data set can be selected by using the "keep" or "drop" commands. Suppose you have a data set with six variables: var1, var2, …, var6. You

would like to keep a file with only three of them (say, var1, var2, and var3). You can use either of the following two commands:

- "keep var1 var2 var3" (or "keep var1-var3" if the variables are in this order)
- "drop var4 var5 var6" (or "drop var4-var6" if the variables are in this order)

Note the use of a hyphen (-) in both commands. It is good practice to use the command that involves fewer variables or less typing (and hence less risk of error). You can also use relational or logical operators. For example, the following command drops those observations where the head of the household is 80 or older:

```
. drop if agehead>=80
```

And this command keeps those observations where household size is six or fewer members:

```
. keep if famsize<=6
```

The preceding two commands drop or keep all variables depending on the conditions. You cannot include a variable list in a "drop" or "keep" command that also uses conditions. For example, the following command will fail:

```
. keep nh famsize if famsize<=6
invalid syntax
r(198)
```

You have to use two commands to do the job:

```
. keep if famsize<=6
. keep nh famsize
```

You can also use the keyword in a "drop" or "keep" command. For example, to drop the first 20 observations:

```
. drop in 1/20
```

Producing Graphs

Stata is quite good at producing basic graphs, although considerable experimentation may be needed to produce beautiful graphs. The following command shows the distribution of the age of the household head in a bar graph (histogram):

```
. histogram agehead
```

In many cases, the easiest way to produce graphs is by using the menus; in this case, click on Graphics and then on Histogram and follow the prompts. An easy way to save a graph is to right-click on it and copy it to paste into a Microsoft Word or Excel document.

Here is a command for a scatterplot of two variables:

```
. twoway (scatter educhead agehead), ytitle(Education of head)
xtitle(Age of head) title(Education by Age)
```

Combining Data Sets

In Stata, only one data file can be worked with at a time—that is, there can be only one data set in memory at a time. However, useful information is often spread across multiple data files that need to be accessed simultaneously. To use such information, Stata has commands that combine those files. Depending on how such information is spread across files, one can *merge* or *append* multiple files.

Merging Data Sets

Merging of data files is done when one needs to use *variables* that are spread over two or more files. As an example of merging, the hh_98.dta will be divided into two data sets in such a way that one contains a variable or variables that the other does not, and then the data sets will be combined (merged) to get the original hh_98.dta back. Open the hh_98.dta file, drop the program participation variables, and save the datafile as hh_98_1.dta.

```
. use hh_98, clear
. drop dmmfd dfmfd
. save hh_98_1.dta,replace
```

You want to give this file a new name (hh_98_1.dta) because you do not want to change the original hh_98.dta permanently. Now open the hh_98.dta again. This time, keep the program participation variables only. Save this file as hh_98_2.dta.

```
. use hh_98, clear
. keep nh dmmfd dfmfd
. save hh_98_2.dta,replace
```

Notice that you kept the household identification ("nh") in addition to the program participation variables. This is necessary because merging requires at least one common identifying variable between the two files that are to be merged. Here "nh" is that common variable between the two files. Now you have two data sets—one has household's program participation variables (hh_98_2.dta), and the other does not have those variables (hh_98_1.dta). If you need to use the variables from both files, you will have to merge the two files. However, before merging two files, you need to make sure that both files are sorted by the identifying variable. This can be done quickly as follows:

```
. use hh_98_1, clear
. sort nh
```

```
. save,replace
. use hh_98_2, clear
. sort nh
. save,replace
```

Now you are ready to merge the two files. One of the files has to be open (it does not matter which file). Open hh_98_1.dta file, and then merge the hh_98_2.dta file with it:

```
. use hh_98_1, clear
. merge nh using hh_98_2
```

In this context, hh_98_1.dta is called the *master file* (the file that remains in the memory before the merging operation) and hh_98_2.dta is called the *using file*. To see how the merge operation went, type the following command:

```
. tab _merge
```

Stata creates this new variable "_merge" during the merging operation. A tab operation to this variable displays different values of "_merge" and thereby status of the merging operation.

```
    _merge |       Freq.     Percent       Cum.
------------+-----------------------------------
         3 |       1129      100.00     100.00
------------+-----------------------------------
     Total |       1129      100.00
```

Even though in this case "_merge" has only one value (3), it can have up to three possible values, depending on the nature of the merging operation:

- A value of 1 shows the number of observations coming from the *master file* only.
- A value of 2 shows the number of observations coming from the *using file* only.
- A value of 3 shows the number of observations common in both files.

The total number of observations in the resulting data set is the sum of these three "_merge" frequencies. In this example, however, each observation (household) in the hh_98_1.dta file has an exact match in the hh_98_2.dta file, which is why you got "_merge=3" and not 1s or 2s (obviously, because the two files are created from the same file). But in real-life examples, 1s and 2s may remain after merging. Most often, one wants to work with the observations that are common in both files (that is, "_merge=3"). That is done by issuing the following command after the merging operation:

```
. keep if _merge==3
```

In addition, it is good practice to drop the "_merge" variable from the data set after the "_merge" operation. Now you have a data set that is identical to hh_98.dta in content.

Appending Data Sets

Appending data sets is necessary when you need to combine two data sets that have the same (or almost the same) variables, but observation units (households, for example) are mutually exclusive. To demonstrate the append operation, you will again divide the hh_98.dta. This time, however, instead of dropping variables, you will drop a few observations. Open the hh_98.dta file, drop observations 1 to 700, and save this file as hh_98_1.dta:

```
. use hh_98, clear
. drop in 1/700
. save hh_98_1.dta,replace
```

Next, reopen hh_98.dta but keep observations 1 to 700, and save this file as hh_98_2.dta.

```
. use hh_98, clear
. keep in 1/700
. save hh_98_2.dta,replace
```

Now, you have two data sets; both have identical variables but different sets of households. In this situation, you need to append two files. Again, one file has to be in memory (which one does not matter). Open hh_98_1.dta, and then append hh_98_2.dta.

```
. use hh_98_1, clear
. append using hh_98_2
```

Note that individual files do not need to be sorted for the append operation, and Stata does not create any new variable like "_merge" after the append operation. You can verify that the append operation executed successfully by issuing the Stata "count" command, which shows the number of observations in the resulting data set, which must be the sum of the observations in the two individual files (that is, 1,129).

Working with .log and .do Files

This section discusses the use of two types of files that are extremely efficient in Stata applications. One stores Stata commands and results for later review (.log files), and the other stores commands for repeated executions later. The two types of files can work interactively, which is very helpful in debugging commands and in getting a good "feel" for the data.

.log Files

One often wants to save the results of Stata commands and perhaps to print them out. Do this by creating a .log file. Such a file is created by issuing a "log using" command

and closed by a "log close" command; all commands issued in between, as well as corresponding output (except graphs) are saved in the .log file. Use hh_98.dta. Assume that you want to save only the education summary of heads by household gender. Here are the commands:

```
. log using educm.log
. by sexhead, sort:sum educhead
. log close
```

What happens here is that Stata creates a text file named *educm.log* in the current folder and saves the summary output in that file. If you want the .log file to be saved in a folder other than the current folder, you can specify the full path of the folder in the .log creation command. You can also use the File option in the Menu, followed by Log and Begin.

If a .log file already exists, you can either replace it with "log using educm.log" or replace or append new output to it with "log using educm.log, append." If you really want to keep the existing .log file unchanged, then you can rename either this file or the file in the .log creation command. If you want to suppress a portion of a .log file, you can issue a "log off" command before that portion, followed by a "log on" command for the portion that you want to save. You have to close a .log file before opening a new one; otherwise, you will get an error message.

.do Files

You have so far seen interactive use of Stata commands, which is useful for debugging commands and getting a good feel for the data. You type one command line each time, and Stata processes that command, displays the result (if any), and waits for the next command. Although this approach has its own benefits, more advanced use of Stata involves executing commands in a batch—that is, commands are grouped and submitted together to Stata instead one at a time.

If you find yourself using the same set of commands repeatedly, you can save those commands in a file and run them together whenever you need them. These command files are called *.do files*; they are the Stata equivalent of macros. You can create .do files at least three ways:

1. Simply type the commands into a text file, label it "educm.do" (the .do suffix is important), and run the file using "do educm" in the Stata Command window.
2. Right-click anywhere in the Review window to save all the commands that were used interactively. The file in which they were saved can be edited, labeled, and used as a .do file.
3. Use Stata's built-in .do editor. It is invoked by clicking on the icon (the fifth from the right, at the top of the page). Commands may then be typed into the editor.

Run these commands by highlighting them and using the appropriate icon (the second from the right) within the .do editor. With practice, this procedure becomes a very quick and convenient way to work with Stata.

Here is an example of a .do file:

```
log using educm.log
use hh_98
sort nh
save, replace
sort sexhead
by sexhead:sum educhead
log close
```

The main advantages of using .do files instead of typing commands line by line are replicability and repeatability. With a .do file, one can replicate results that were worked on weeks or months before. Moreover, .do files are especially useful when sets of commands need to be repeated—for instance, with different data sets or groups.

Certain commands are useful in a .do file. They are discussed from the following sample .do file:

```
*This is a Stata comment that is not executed
/*****This is a do file that shows some very useful
 commands used in do files. In addition, it creates a
log file and uses some basic Stata commands  ***/

#delimit ;
set more 1;
drop _all;
cap log close;
log using c:\eval\log\try.log, replace;

use c:\eval\data\hh_98.dta ;
describe ;
list in 1/3 ;
list nh famsize educhead if sexhead==0 & agehead<45;
summarize famsize;
summarize famsize, detail;
sum famsize educhead [aw=weight], d;
tab sexhead;
tab educhead sexhead, col row;
tab educhead, summarize(agehead);
label define sexlabel 1 “MALE” 0 “FEMALE”;
label values sexhead sexlabel;
tabulate sexhead;
```

```
label variable sexhead "Gender of Head: 1=M, 0=F";
save c:\eval\data\temp.dta, replace;
#delimit cr
use c:\eval\data\hh_91.dta
append using temp
tab year
log close
```

The first line in the file is a comment. Stata treats any line that starts with an asterisk (*) as a comment and ignores it. You can write a multiline comment by using a forward slash and an asterisk (/*) as the start of the comment, and end the comment with an asterisk and forward slash (*/). Comments are very useful for documentation purposes, and you should include at least the following information in the comment of a .do file: the general purpose of the .do file and the last modification time and date. You can include comments anywhere in the .do file, not just at the beginning.

Commands used in the sample .do file are as follows:

`#delimit;` — By default, Stata assumes that each command is ended by the carriage return (that is, by pressing the Enter key). If, however, a command is too long to fit on one line, you can spread it over more than one line. You do that by letting Stata know what the command delimiter is. The command in the example says that a semicolon (;) ends a command. Every command following the "delimit" command has to end with a semicolon. Although for this particular .do file the "#delimit" command is not needed (all commands are short enough), it is done to explain the command.

`set more 1` — Stata usually displays results one screen at a time and waits for the user to press any key. But this process would soon become a nuisance if, after letting a .do file run, you have to press a key for every screen until the program ends. This command displays the whole output, skipping page after page automatically.

`drop _all` — This command clears the memory.

`cap log close` — This command closes any open .log file. If no log file is open, Stata just ignores this command.

Exercise: Run the .do file sample.do, which stores its output in try.log. When you see "end of .do file," open c:\eval\log\try.log in Microsoft Word (or Notepad) and check the results.

.ado Files

The .ado files are Stata programs meant to perform specific tasks. Many Stata commands are implemented as .ado files (for example, the "summarize" command). To run such a program, simply type the name of the program at the command line. Users can write their own .ado programs to meet special requirements. In fact, Stata users and developers are continuously writing such programs, which are often made available to the greater Stata user community on the Internet. You will use such commands throughout the exercises on different impact evaluation techniques. Stata has built-in commands to download and incorporate such commands in Stata. For example, the propensity score matching technique is implemented by an .ado file called *pscore.ado*. To download the latest version of this command, type the following command at the command line:

```
. findit pscore
```

Stata responds with a list of .ado implementations of the program. Clicking on one of them will give its details and present the option to install it. When Stata installs an .ado program, it also installs the help files associated with it.

Follow-up Practice

Look at the 1998 data set that will be used frequently in the impact evaluation exercise.

a. Household characteristics

Look at how different the household characteristics are between the participants and nonparticipants of microfinance programs. Open c:\eval\data\hh_98.dta, which consists of household-level variables. Fill in the following table. You may use the "tabstat" or "table" command in Stata.

```
. tabstat famsize, statistics(mean sd) by(dfmfd)
. table dfmfd, contents(mean famsize sd famsize)
```

	Full sample		Female participants		Households without female participants	
	Mean	Standard deviation	Mean	Standard deviation	Mean	Standard deviation
Average household size	_____	_____	_____	_____	_____	_____
Average household assets	_____	_____	_____	_____	_____	_____
Average household landholding	_____	_____	_____	_____	_____	_____
Average age of household head	_____	_____	_____	_____	_____	_____
Average years of education of household head	_____	_____	_____	_____	_____	_____
Percentage of households with male head	_____	_____	_____	_____	_____	_____

Are the sampled households very different among the full sample, participants, and nonparticipants?

Gender of household heads may also affect household characteristics.

```
. tabstat famsize, statistics(mean sd) by(sexhead)
. table sexhead, contents(mean famsize sd famsize)
```

	Male-headed households		Female-headed households	
	Mean	Standard deviation	Mean	Standard deviation
Average household size				
Average years of head schooling				
Average head age				
Average household assets				
Average household landholding				

Are the sampled households headed by males very different from those headed by females?

b. Village characteristics

	Mean	Standard deviation
If village is accessible by road		
Percentage of village land irrigated		

c. Prices

	Full sample		Participants		Nonparticipants	
	Mean	Standard deviation	Mean	Standard deviation	Mean	Standard deviation
Rice						
Wheat						
Edible oil						
Milk						
Potato						

d. Expenditure

Open c:\eval\data\hh_98.dta. It has household-level consumption expenditure information. Look at the consumption patterns.

	Per capita expenditure		Per capita food expenditure		Per capita nonfood expenditure	
	Mean	Standard deviation	Mean	Standard deviation	Mean	Standard deviation
By head gender						
Male-headed households						
Female-headed households						
By head education level						
Head has some education						
Head has no education						
By household size						
Large household (> 5)						
Small household (<= 5)						
By land ownership						
Large land ownership (> 50/person)						
Small land ownership or landless						

	Full sample		Female participants		Households without female participants	
	Mean	Standard deviation	Mean	Standard deviation	Mean	Standard deviation
Per capita expenditure						
Per capita food expenditure						
Per capita nonfood expenditure						

Summarize your findings on per capita expenditure comparison.