Lab 1: Basics of Stata
Short Course on Poverty & Development for Nordic Ph.D. Students
University of Copenhagen
June 13-23, 2000

This lab is designed to give you a basic understanding of the tools available in the statistical package called Stata (pronounced "stay-tuh"). Stata has many built in commands to do such things as simple regression, two- and three-stage least squares, probit/logit/tobit models, Heckman selection models (maximum likelihood and two-stage methods), and many more. As such, most of an analyst's time spent using this program involves manipulation of data and output from the various procedures. Of course, you can also write your own procedures, as we will do in successive labs. But for now, we'll concentrate on the basics of data managment. The outline for this lab is as follows:

A. Data Management and Transfer
B. Loading, Saving and Merging Datasets
C. Data Modification and Editing
D. Basic Descriptive Analysis
E. Basic Regression Analysis
F. Submitting batch files (i.e. '.do' files)

Because we do not have access to computers for this first lab, this sheet is designed more as a reference than as an exercise (unlike the remaining labs).

A) DATA MANAGEMENT AND TRANSFER

A.1) Typing data into the Stata Editor

```
. clear
```

The clear command clears out the dataset that is currently in memory. We need to do this before we can create or read a new dataset.

Example: A small dataset

name	midterm	final
Smith	79	84
Jones	87	86
Brown	91	94
Adraktas	80	84

Creating a dataset using the 'Data Editor'

```
. edit
```

The edit command opens up a spreadsheet like window in which you can enter and change data. You can also get to the 'Data Editor' from the pull-down 'Window' menu or by clicking on the 'Data Editor' icon on the tool bar.

Enter values and press return. Double click on the column head and you can change the name of the variables. When you are done click the 'close box' for the 'Data Editor' window.

```
. save class91
. save class91, replace
```

The save command will save the dataset as class91.dta. Editing the dataset changes data in the computer's memory, it does not change the data that is stored on the computer's disk. The replace option allows you to save a changed file to the disk, replacing the original file.

A.2) Reading Data

We are going to see the following commands: use, insheet, infile, infix dictionary files and using conversion programs.

.Use

The Stata "use" command reads data that has been saved in Stata format:

```
. use myfile
```

where "myfile" is the name of the Stata file. Note that the ".dta" file extension is automatically appended to Stata files. You do not have to include the file extension on the use command.
On a PC, use the "File/Open..." menu for reading Stata formatted data. Stata will automatically clear your memory from any existing data.

.Insheet

The insheet command is used to read data from a file created by a spreadsheet or database program. The raw values in the file must be either comma or tab delimited (we will generically refer to raw data as "filename.raw").

- Comma/Tab separated file with variable names on line 1:

The names are included in the file. There are some other conventions required:

1.The first line should have Stata variable names (eight characters or less, not starting with a "special character" or number) and the second line begins the data.

2.Missing numeric data should be coded as an empty cell, not a space, dot, or any other non-numeric data. Often, 0, 9, or 99 is used to code missing numeric data; this is fine as long as these are not also valid values for that variable.

3.Commas in numbers or text are particularly problematic because Stata thinks they are a delimiter and will not read the data properly. You must remove the commas from numeric values before saving the file.

4.When some spreadsheets create a .csv file it does not add commas to the end of a line if the cells at the end of that line are empty. This will confuse Stata, which relies on the commas to tell it where the values are. You can avoid this problem by adding another column of 1's (or any other character) to your spreadsheet. You can drop this variable once you have read it into Stata.

5.The file must be specifically saved as a "comma separated values" file in Excel. You can do this by going to "File", then "Save As…", then choosing "comma separated values." Simply giving it an extension of ".csv" will not work.

When you close the spreadsheet and Excel asks if you want to save the changes, say "No." This is counter-intuitive, but the changes it's asking you about are the changes it needs to make the spreadsheet a regular Excel spreadsheet again.

Once you have completed the changes and you are in Stata, give the command:

```
. insheet using filename.csv
```

to read in the file. If you get an error message about "wrong number of values," then you have a problem with not enough commas in the file, see #4 above.

Example:
Type in Excel:

name	midterm	final
Smith	79	84
Jones	87	86
Brown	91	94
Adraktas	80	84

Save as "Spread" using the instructions above. We have a spreadsheet file called spread.csv that looks like this:

name,midterm,final
Smith,79,84
Jones,87,86
Brown,91,94
Adraktas,80,84

Then:

```
. clear
. type spread.csv
. insheet using spread.csv
. list
```

- Comma/Tab separated file (with no variable names in file):

Everything works as before, only that now, the insheet command includes the variable names:

```
. insheet name midterm final using filename.csv
```

.Infile

The infile command is used to read data from an external ascii file, which conforms to these specifications:

1. The file should NOT have variables names on the first line.
2. Character variables that have spaces in them, such as full names, must be enclosed in quotes.
3. Numbers can have commas and minus signs, but not dollar or percent signs.
4. Infile assumes that the variables have spaces between them and that there are no blanks spaces where it expects data (missing data needs to be represented by something).

The command to read a file with infile is:

```
. infile var1 var2 var3 using mydata.raw
```

In this case we also need to tell Stata when a variable is a character variable, and how long it can be.

Example:
We have an ASCII file called ascii.raw that looks like this:

```
"A. Smith"      79      84
"Jones"      87      86
"Brown"      91      94
"Adraktas"   80      84
```

Lets read it:

```
. clear
. type ascii.raw
. infile str10 name midterm final using ascii.raw
. list
```

The names of the variables are given followed by the keyword using which in turn is followed by the name of the file. str10 is not a variable name but indicates that name is a string variable up to 10 characters long.
(It may be easier, though, to bring a file into Excel first, save it as a csv, then use insheet).

. Infix
Used for fixed format files. See Stata Users Guide.

. Dictionary Files
Used when there is no specific format. See Stata Users Guide.

. Using data conversion programs.
Some programs can convert data from one file format to another file format. For example, they could directly create a Stata file from an Excel Spreadsheet, a Lotus spreadsheet, an Access database, a Dbase database, a SAS data file, an SPSS system file, etc… Two such examples are Stat Transfer and DBMS Copy.

B) LOADING, SAVING, MERGING DATA SETS

B.1) Loading Stata datasets

You will load data with the “use” command, once the data are in Stata format. Often, your dataset will be larger than the default and you will need to increase the amount of memory Stata uses. To do this, use the "set memory" command:

```
. set mem 10m
. use [filename]
```

This example increases the memory to 10 megabytes and then loads the data set. Make sure that you give yourself some extra memory so you can create new variables and/or add observations.

Some basic commands to obtain information about our dataset are:

```
. Describe
```

This command displays a summary of a Stata dataset, describing the variables, their labels and other information.

```
. List
```

It is often useful to just look at the data without doing any kind of analysis. The "list" command, abbreviated as "l" will let you do this. Simply giving the "l" command will display all of the data on your screen, if you specify certain variables or observations in the command, then only those variables or observations will be printed.

Example:

```
. list var1 var2 in 1/10
```

Only the first 10 observations of variables 1 and 2 will be printed.

```
. Codebook
```

Displays information about variables’ names, labels and values.

B.2) How to Save Stata Files

You can use the Stata save command to save a file in Stata format:

```
. save myfile
```

where “myfile” is the name of your Stata file. It will be saved in your working directory. This file can be read in Stata with the use command.

If you already have a Stata file named "myfile.dta" and wish to save an updated version of the file under the same name:

```
. save myfile, replace
```

To save an updated version of the active file, you can simply type:

```
. save, replace
```

In Stata for PC, to save the file "myfile.dta" use the "File/Save As..." menu.

B.3) Merging data sets

Once you have all of your variables in the format you want, you'll need to get the entire file in a format that will make it easier to use. You can do this by sorting, appending, merging and collapsing.

. Sort

"sort" puts the observations in a data set in a specific order. Some procedures require the file to be sorted before it can work.
Creating the id variable (identification variable) is important. With the id variable, you will always be able to go back to the original order and start over.

```
. sort id
```

In Stata for PC you can sort data in the Stata Editor, by clicking on the variable you want to sort by, and then clicking the "Sort" button. :

. Append

Sometimes, you have more than one file of data that you need to analyze. One case may be that you have two files with the same variables but different observations. The other case is when you have two files with the same observations, but different variables.

Use append when you simply want to add more observations, in other words you already have data for 1990, and now you want to include new observations from 1991 for the same variables. For example:

```
. use class91
. list
        name   midterm     final   examsep
  1.     Huang        95        80         .
  2.    Dupont        72        76         .
  3.     Perez        68        64        78


. use class90
. list
         name   midterm     final
  1.     Smith        79        84
  2.     Jones        87        86
  3.     Brown        91        94
  4.  Adraktas        80        84


. append using class91

. list

         name   midterm     final   examsep
  1.     Smith        79        84         .
  2.     Jones        87        86         .
```

```
  3.     Brown        91        94         .
  4.  Adraktas        80        84         .
  5.     Huang        95        80         .
  6.    Dupont        72        76         .
  7.     Perez        68        64        78
```

This command will add the observations from the file, class91 (what Stata refers to as the "using" dataset), to the end of class90 (what Stata refers to as the "master" dataset). Any variables with different names in the two files will have missing values for the observations from the other dataset.

. Merge

If you have two files that have the same observations, but different variables, then you'll want to "merge" them so you can use all of the variables at once. When you merge datasets, you are adding new variables to existing observations rather than adding observations to existing variables. There are two basic kinds of merges, a one-to-one and a match.

A one-to-one merge simply takes the two files and puts them side-by-side, regardless of whether the observations in each dataset are in the same order. A simple one-to-one merge isn't very common, and isn't recommended, even if you think all the observations match.
There are some "rules" when doing a match-merge. First, each dataset must have a "key" variable by which the observations can be matched – id variable is a good example. Second, both datasets must be sorted by this key variable. You can use more than one key variable if you want, such as month and year. These key variables must be of the same type (string or numeric) in both datasets. By default, if a variable is present in both datasets, then the values in the master dataset will remain unchanged. If the variables in the using dataset have the same names as ones in the master dataset, but represent additional information, then you will need to rename them in one of the datasets before you can merge them.

Stata will automatically create a variable called _merge which will indicate the results of the merge. Always check this variable to make sure that you got what you wanted. Here are what the possible values of _merge mean:

1 = Observations from the master dataset that did not match observations from the using dataset.
2 = Observations from the using dataset that did not match observations from the master dataset.
3 = Observations from both datasets that did match.

Usually, you will want all the observations to have a value of 3.

If you need to merge another dataset, you have to re-name or drop _merge first, other wise you will get a "_merge already defined" error. :

. Collapse

"collapse" is used when you want to create a dataset containing the means sums, etc., of the various groups in the data. One example might be when you have one dataset of monthly data and another of yearly data and you need to analyze both sets of information together.

```
. collapse (mean) var1 var2, by(month)
```

This will create a dataset of the means of var1 and var2 by month. If you have twelve months in your original dataset, then you will have twelve observations in the collapsed dataset. You must be careful, though, because Stata will compute the statistics on a variable-by-variable basis. If one variable has more missing observations than another, the means (and any other statistics you request) will be based on a different number of observations. This is not always an acceptable practice. To avoid this, you must use the "cw" options for "casewise deletion." This means that Stata will drop any observation that does not have data for both var1 AND var2, thereby ensuring that all the statistics will be based on the same number of observations.

```
. collapse (mean) var1 var2, by(month) cw
```

Example: In a data set by individuals, we can obtain information for households doing, for example:

```
. collapse (sum) var1 var2 by (hhid)
```

C) DATA MODIFICATION AND EDITING

C.1) Creating Variables

Stata can store data as either numbers or characters. Stata will allow you to do most analyses only on numeric data.

There are different types of numeric variables - float, binary, double, long and int - the differences among them are simply how much space they take up in the file. In most cases, you will not need to concern yourself with these differences.

. Generate (gen)

The "gen … real" command will create a numeric variable based on the original string variable. Use this if the original was defined as a character variable "by mistake." The following command will create a new, numeric variable var1n by converting the string variable, var1:

```
. gen var1n=real(var1)
```

Often, you will need to create new variables based on the ones you have already. The two most common ways of creating new variables is by using "generate" and "egen." Here are some examples of gen:

```
. gen total= var1n + var2 + var3
. gen cumtot=sum(total)
```

In the first example, we generate a new variable called "total" which is simply the addition of var1, var2, and var3. In the next example, we create a cumulative sum of total (the value of cumtot for this observation is the sum of total for all previous observations).

Logical operators used in Stata

~ not
= = equal
~ = not equal
! = not equal
> greater than
> = greater than or equal
< less than
< = less than or equal
& and
| or

Sometimes we need to generate a "dummy" variable, or variables. Stata makes this very easy:

```
. tab var1, gen(dummy)
```

Here, Stata will create a dummy variable for each value found in var1.
Stata assumes that the variable you are creating is numeric unless you tell it otherwise. Sometimes, though, you will want to create a variable whose values are strings. Here's how:

```
. gen str5 name="John"
. gen str10 team="Atlanta" in 1/5
. gen str3 yn="yes" if team=="Atlanta" & name=="john"
. replace yn="no" if team=="New York" | team=="Boston"
```

. Replace
The replace command allows you to change an existing variable.

. Extended Generate (egen)
"egen", or "extended generate" is useful when you need a new variable that is the mean, median, etc. of another variable, for all observations or for groups of observations. Egen is also useful when you need to simply number groups of observations based on some classification variables. Here are some examples:

```
. egen sumvar1 = sum(var1)
. egen meanvar1= mean(var1), by(var3)
. egen count = count(id), by(company)
```

```
. egen group = group(month year)
```

There are many functions that can be used with "extended generate". To have a brief summary, type:

```
. help egen
```

C.2) Variable and Value labels

Variable labels correspond to the variable names, whereas value labels correspond to the different values a variable may have. Here's how to create them:

```
. label variable var1 "The first variable"
```

This assigns a label to the variable, var1.

Often, though, it's more helpful to label the values a variable can take so you don't have to memorize them or keep referring to a codebook.

```
. label define grp 1 "Male" 2 "Female" 3 "N/A"

. label values gender grp
```

First you have to "define" the label, then associate that label with a variable.

C.3) Keep, Drop, and Rename

We can either keep the variable we are interested in:

```
. keep code date var4
```

or we can drop those we don't need, and rename some of those we keep:

```
. drop var1 var2 var3
. rename code date
```

Be careful! Once they are dropped, they can't be picked up again unless you clear the data and use it again.

You can also drop (or keep) observations:

```
. drop if size==1
. keep if size~=1
```

D) BASIC DESCRIPTIVE ANALYSIS

Summarize

"sum", short for summarize, will give you the means, sd's, etc. of the variables listed. If you don't list any variables, it will give you the information for all numeric variables. If a variable you thought was numeric shows up as having 0 observations and a mean of 0, then, most likely, Stata still thinks it's a character variable.

```
. sum var1 var2
```

The "detail" option gives you additional information about the distribution of the variable.

```
. summ var1 , detail
```

Tabulate
"tab", short for tabulate, will produce frequency tables. By specifying two variables, you will get a crosstab. There are other options to get the row, column and cell percentages as well as chi-square and other statistics;

```
. tabulate varrow varcolumn [, cell column row missing nofreq
wrap nolabel all chi2 exact gamma lrchi2 taub V]
```

Type “help tabulate” for more details.

Sometimes you’ll want to run a command or analysis on different groups of observations. The "by variable:" subcommand is the same thing as running the command with separate "if" statements for each group. You must sort the data before you can use the "by:"

```
. sort urban
. by urban: tab var1 var2, row col
```

Histograms
Basic histograms can be obtained with the *graph* command.

Use of Weights
We usually have to indicate the weight to be attached to each observation. The syntax of weight is [weightword=exp]
Possible weightwords:

- weight: default treatment of weights
- aweight: analytic weights
- fweight: frequency weights
- pweight: sampling weights
- iweight : importance weights

E) BASIC REGRESSION ANALYSIS

. Regress
In Stata you can estimate regression models using the "regress" command (shortcut reg).
Suppose your dependent variable is depvar and your independent variables are listed in varlist. Then the command to run the linear regression is:

```
. reg depvar varlist
```

The first variable after the regress command is always the dependent variable (or left-hand-side variable), and the following list gives the relevant independent variables.

Example: you have a data set with individual observations on earnings, and you are interested in regressing the natural logarithm of average weekly earnings (lnw) on a sex dummy variable, on a race dummy variable, on years of schooling, on age and on a constant. Then you should type:

```
. reg lnw sex race school age
```

Notice that Stata automatically adds the "constant" (_cons) to the list of independent variables. If you want to exclude it, you can use the option nocons

```
. reg lnw sex race school age, nocons
```

Suppose that you want to run a regression only for the male population:

```
. reg lnw race school age if sex==0
```

Several options are available with the regress command (see help regress). Among the most useful is the robust option, that allows you to run the regression using a "robust" estimator for the variance (specifically, the Huber/White/sandwich estimator). This option is mainly used when the residuals are thought to be heteroschedastic:

```
. reg lnw sex race school age, robust
```

Another option that may be useful is the one that allows you to run Instrumental Variable regressions (or 2SLS). Suppose you have a dependent variable y1 and a list of independent variables x1 x2 x3; and let's suppose that you want to instrument x3 with z1, x1 and x2. Then you can simply run the (2SLS) regression by typing:

```
. reg y1 x1 x2 x3 (z1 x1 x2)
```

and you will get the results of the second stage in the 2SLS procedure.

. Logit, Probit

When you dependent variable is dichotomous (zero-one), you can run a Linear Probability Model (using the **regress** command) or you might be interested in running a Logit or a Probit regression.

To run a Logit regression in Stata, you simply type:

```
. logit depvar varlist
```

where *depvar* is your dependent (dichotomous) variable, and *varlist* is the list of independent variables that enter the model. To run a Probit model, the command is similar:

```
. probit depvar varlist
```

Stata reports the maximum likelihood estimates for the original coefficients in both the *probit* and *logit* commands. You might be interested in other "coefficients". The *dprobit* command estimates a Probit model, but reports the change in the probability for an infinitesimal change in each independent variable instead of the coefficient for each independent variable. The *logistic* command estimates a Logit model but reports the odds ratios for each independent variable instead of the coefficients. The *logistic* command also offers several "post-estimation" commands that can be used to evaluate the goodness of fit of the model. Type `help logistic` for a list.

Stata also saves various results as matrices (vectors are considered matrices in Stata) and scalars. See the Reference manuals for saved results.

F) SUBMITTING BATCH FILES (i.e. '.do' files)

While we can submit commands to Stata one-by-one interactively, we can also submit a bunch of commands in the form of batch files, or in Stata-speak, '.do' files. This is very useful when (if you are at all like me) you make a coding error early in a series of commands. If you submit the commands as a '.do' file, then you need only go back to the file, change the coding mistake, and resubmit the commands.

The way it works is that you create a text file using your favorite text editor (e.g. Notepad) and save it with the extension '.do' (e.g. c:\pathname\myprog.do). You enter the commands in this file, save it, open up Stata, and from the Stata Command window type `do c:\pathname\myprog.do`. You can also change the directory to the one in which your file is located (e.g. type `cd c:\pathname`) and then type `do myprog.do`.

It is generally a good idea to create a log file that records all of the output that appears on the screen. AND, it's generally a good idea to give it the same name as your '.do' file, but give it a '.log' extension. Here is an example of what a '.do' file looks like (you'll see more in successive labs):

```
version 6.0
set memory 5m
#delimit ;

log using c:\pathname\myprog.log, replace

* * * * * * * * * * * * * * * * * * * *
  Describe what the program does
* * * * * * * * * * * * * * * * * * * *;

use c:\datalocation\mydata.dta;

 -- commands . . ;

log close;
```

~~~~~~

For those of you who are Danes, the following e-mail posted on the Statalist Serv might be of interest:

```
I developed an introductory note in danish for Stata. The note shows
different ways of working with Stata, menubased (quest/labedit etc), do
file editor based and a combination of external editors and stata. The note
is available on internet as pdf file on:

complete note: www.bola.suite.dk/stata/stintro.pdf
contents:    www.bola.suite.dk/stata/statanot.pdf
```
~~~~~~

The note contains 78 pages and 85 exercises. To run the exercises download datafiles and supplementary ado files as well: available from same adress.

Kindly Jens M.Lauritsen JM.Lauritsen@dadlnet.dk