

STATA I: Beginning STATA or IS STATA REALLY AS FRIENDLY AS THEY SAY?

Anita Rocha
Center for Studies in Demography and Ecology
University of Washington
Revised 4/25/2007

Table of contents

| | |
|---|----|
| Introduction..... | 1 |
| Overview..... | 2 |
| Basic Screen Navigation..... | 2 |
| Do-file Editor..... | 3 |
| Import Data..... | 4 |
| Comments..... | 5 |
| View Data..... | 6 |
| Descriptive Statistics..... | 7 |
| Labels..... | 11 |
| Dropping Variables or Observations..... | 13 |
| Creating New Variables..... | 13 |
| Regression..... | 15 |
| Graphics..... | 17 |
| Merging Datasets..... | 19 |
| Reshape Data..... | 20 |

Introduction

This is a course on how to begin using the statistical programming package Stata® developed by StataCorp (www.stata.com), specifically Stata version 9. This course assumes no previous experience with Stata. However, prospective students should be familiar with fundamental statistical concepts and the Windows operating environment.

Stata can be used for data management, statistical analysis, and presentation of results.

First, here are some useful resources.

| | |
|-------------------------------|---|
| Carolina Population Center | http://www.cpc.unc.edu/services/computer/presentations/statatutorial/tutorial |
| UCLA STATA resources | http://www.ats.ucla.edu/stat/stata/ |
| STATA NetCourses | http://www.stata.com/netcourse/ |

Overview

Stata is a versatile tool. We will look at just a small part, the part that can be useful to demographic researchers. But there's a whole lot more.

Why use STATA?

- 1) Intuitive data management capabilities. The creation of variables and sub-setting data is simple and straightforward.
- 2) Wide variety of statistical procedure that can be accessed via a point-and-click method.
- 3) Syntax is provided so users can learn to code quickly.
- 4) Users can share complex coding syntax with others.

Possible weaknesses

- 1) Not always easy to handle large datasets.
- 2) Documentation can be sparse for those who want more detail.
- 3) Complex programming can become challenging.

There are other packages that have strengths and weaknesses compared to Stata. If you'd like to compare Stata, SAS, and SPSS statistical packages, consider reading the technical report titled *Strategically using General Purpose Statistics Packages: A Look at Stata, SAS and SPSS* provided on the UCLA Academic Technology Services site (http://www.ats.ucla.edu/stat/technicalreports/Number1/ucla_ATSstat_tr1_1.0.pdf)

Topics for this workshop will include

- Navigating Stata windows
- Importing and Saving Stata data files
- Describing data
- Using the documentation and point-and-click methods
- Sub-setting data
- Creating new variables and changing values of existing variables
- Regression
- Graphics
- Merging datasets and reshaping

Basic Screen Navigation

Open Stata version 9.

Main windows:

- 1) Results – output and messages about your Stata code (like errors)
- 2) Command – place to type in code

Side windows:

- 3) Review – a record of all the code typed into the command box
- 4) Variables – list of variables in the data in memory

You can adjust window sizes by the drag n' drop method.

Q: What if you'd like to recover the default settings?

A: Click on the top drop-down menu as follows.

Prefs ► Manage Preferences ► Load Preferences ► Factory Settings

Exercise 1:

What happens when you change the push pin symbol at the top right corner of the Command, Review or Variable windows? How do you get them back to their original positions?

What if you click on the × at the top right corner of the Review or Variable windows? How do you get the window back?

(Hint: see top drop-down menu Window.)

Do-file Editor

A do-file editor is an optional window and is especially useful if you intend to keep a record of your code for, say, back-up, checking, and altering. I've found that most researchers need to save a file with their "draft" and "final" code that accomplishes key data manipulation and analysis.

Open the Do-file Editor by clicking on the top drop-down menu as follows.

Windows ► Do-file Editor ► New Do-file

This window is where a user can run and save code. (Tony Perez has offered and may again offer an extremely useful workshop showing the symbiotic relationship between Stata and Textpad. Textpad, with some distinct advantages, can be another location where a user can run and store Stata code.)

When you save a file in the Do-file Editor, it creates a "do-file", a file with an extension .do. That's the default extension for files containing Stata code.

Exercise 2:

Save the (as of now empty) contents of your do-file in the Do-file Editor as follows.

File ► Save as

Be sure to navigate to your temporary folder on C drive and then give a descriptive name. It will automatically have the .do extension added.

Import Data

Example Data: sub-sample of MORG

The data used for this workshop comes from the Merged Outgoing Rotation Groups (MORG), otherwise known as the U.S. Census Bureau's Current Population Surveys (CPS) Annual Earnings File compiled by the National Bureau of Economic Research (NBER). Fifty or more variables each month are selected for continuity across years. There are 25,000 records or more per month. Here we're using just a sub-sample of the full 2005 dataset for illustrative purposes. For more information and to gain access to more MORG data, go to the CSDE website (<http://csde.washington.edu/>) or email Anita Rocha (alrocha@u.washington.edu).

Sub-sample contents:

| | |
|----------|--|
| AGE | Age |
| EARNHRE | Earnings per hour in pennies |
| EARNWKE | Earnings per week |
| HHID | Household ID; unique for each record in this sub-sample |
| HOURSLW | Number of hours worked last week |
| IHIGRDC | Imputed highest grade completed |
| LNAME | Last name (this is not in the original dataset; it has been added by me to give us an extra TEXT string variable) |
| MARITAL | Marital status 1 = Married, Civilian Spouse Present 2 = Married, Armed Forces Spouse Present 3 = Married, Spouse Absent (exc. Separated) 4 = Widowed 5 = Divorced 6 = Separated 7 = Never Married |
| OWNCHILD | Number of biologic children < 18 years old |
| SEX | Sex 1 = Male 2 = Female |
| YRCOLL | Years of college credit |

Let's begin by importing a sub-sample of the MORG data into Stata. Stata datasets have the extension **.dta**.

Exercise 3:

First, in what folder does your Stata dataset **morg05s1.dta** reside? We are going to open this data file three different ways in STATA.

1) Drop-down menu: from the top drop-down menu, click on the following.

File ► Open ► *navigate to your dataset, click on it, and click on Open*

What changes in your Review and Variable windows?

2) Command line: type the following code into the Command window and then hit Enter.

```
use "C:\{your-path}\morg05s1.dta", clear
```

What's the option **clear** mean? **Clear** specifies that it is okay to replace the data in memory, even though the current data has not been saved to disk.

3) Do-file Editor: click once on the Review window's entry starting with **use ...** From the Command window copy and paste the code into the Do-file Editor window. Save the contents of the file again. (Hint: top drop-down menu File ► Save.) Highlight the command **use ...** and click on the menu-item **do-current file**. (It looks like a paper with writing on it with a down arrow.)

What does “do” and “run” mean in STATA? **Do** and **run** cause STATA to execute the commands stored in filename just as if they were entered from the keyboard. **Do** “echoes” (i.e. shows) the commands as it executes them, whereas **run** is “silent” (i.e. does not show commands).

Yes, STATA is case sensitive! You must type **use** and not **USE** or **Use**.

Note that if you click on a variable in the Variables window, it automatically appears in the Command window. The same is true for a single line of code in the Review window.

Comments

There are four different ways to insert a comment (non-executed text) into STATA code.

| Type of comment | .do .ado files | Command window |
|---|----------------------|-------------------|
| 1. begin the line with * * This line is ignored | √ | √ |
| 2. place the comment between /* and */ delimiters code /* This part of the line is ignored*/ code | √ | |
| 3. begin the comment with // code // To the end of the line is ignored | √ | |
| 4. begin the comment with /// code /// To the end plus carriage return is ignored | √ | |

View Data

Now that we have the data in STATA's memory, let's look at it.

Exercise 4:

View the meta-data (the data about the data) of your dataset. To view the "contents" of the dataset, type the following into the Command window.

describe

How many observations (obs) are in this dataset?
How many variables (vars) are in this dataset?

To see more information about the variables, use command **codebook**.

codebook

To see even more information, use command **inspect**.

inspect

What do **datatypes** mean?

Numeric datatypes

| Storage type | Minimum | Maximum | Closest to zero without being zero | Bytes |
|--------------|----------------------------------|---------------------------------|------------------------------------|-------|
| byte | -127 | 100 | +/-1 | 1 |
| int | -32,767 | 32,740 | +/-1 | 2 |
| long | -2,147,483,647 | 2,147,483,620 | +/-1 | 4 |
| float | $-1.70141173319 \times 10^{-38}$ | $1.70141173319 \times 10^{-36}$ | $+/-10^{-36}$ | 4 |
| double | $-8.9884656743 \times 10^{-307}$ | $8.9884656743 \times 10^{307}$ | $+/-10^{-323}$ | 8 |

String (text) datatypes

| String storage type | Maximum length | Bytes |
|---------------------|----------------|-------|
| str1 | 1 | 1 |
| str2 | 2 | 2 |
| ... | | |
| ... | | |
| ... | | |
| str244 | 244 | 244 |

Review the results from the **describe** command to see what datatypes are included in this dataset.

Exercise 5:

View the actual data. Type the following into the Command window.

list

Just by looking at the data, why would IHIGRDC be stored as a float?
What do missing values look like for string (text) variables?
What do missing values look like for numeric variables?

Exercise 6:

View a partial list of the data by selecting a set of variables (lname, sex and age) and only a portion of the observations (the first ten observations). Type the following into the Command window.

```
list lname sex age in 1/10
```

Note: the syntax **in 1/10** selects the observations 1 through 10.

Exercise 7:

Who are the youngest three people in this sub-sample? First type the code

```
sort age
```

to sort the dataset by AGE and then list the top three observations.

Exercise 8:

Who are the oldest five people in this sub-sample? Type the code

```
list lname age in -5/-1
```

Note that **in -5/-1** goes from the fifth from the bottom to the first from the bottom of the list.

Descriptive Statistics

To begin exploring data, you need to find ways to view it in the aggregate and that is the role of descriptive statistics.

Exercise 9:

What is the average earnings per hour in this data?

```
summarize earnhre
```

What are the minimum and maximum earnings per hour?

Note: if you entered **summarize** by itself, it would provide descriptive statistics for all

the variables in your dataset. You can also use the shortened keyword **sum** instead of **summarize**.

One of STATA's most useful features is the ability to attach an **if** or **by** statement to just about any command syntax. This allows a user to subset the data on-the-fly.

Exercise 10:

A) What is the average earnings per hour in this data when the highest grade completed is over 15 years?

```
summarize earnhre if ihigrdc > 15
```

B) What is the average earnings per hour when the highest grade completed is 12 or fewer years?

C) What is the average earnings per hour for females?

(Hint: use two equal signs (= =) to indicate equivalence within the if-clause.)

D) What is the median earnings per hour? Try the code below with option **detail**.

```
summarize earnhre, detail
```

E) What is the average earnings per hour by *each* level of highest grade completed?

```
sort ihigrdc  
by ihigrdc: summarize earnhre
```

Note the use of SORT before the BY statement of the same variable.

This is a good time to start looking at the Stata documentation. It may take a little getting used to, but this it's an extremely useful tool once you get familiar with it.

Exercise 10A:

What are some of the features of the **summarize** command?

From the drop-down menu

Help ► Search

Type **summarize** in the Keyword Search box. Locate the command **summarize** among the search results. Now, what does the **summarize** option **separator** do? How might you learn about what values you could use for **separator**?

Exercise 11:

What is the correlation between AGE and EARNHRE?

```
correlate age earnhre
```

Now do this only for females using an **if**.

Exercise 11A:

Is the correlation between AGE and EARNHRE statistically significant?

```
pwcorr age earnhre, obs sig
```

What did the options **obs** and **sig** accomplish? If you couldn't tell by the results, how could you check?

The **if** and **by** statements can be used in many different settings. Another way is to use the **by** as an option in a t-test.

Exercise 12:

Perform a t-test of the average earnings per hour comparing females to males.

```
ttest earnhre, by(sex)
```

Note that sorting is unnecessary with the t-test command.

Analysis: Based on these data and assuming a two-sided hypothesis, we are unable to draw any conclusions about the difference between the average earnings per hour for males and females.

What is the numeric relationship between missing values and valid numbers? Try an **if** statement to show the distinction.

Exercise 13:

In STATA a numeric missing value is *larger* than all numbers (which is a bit counter-intuitive). So if a user wants to list observations that are non-missing, then identify all values that are smaller than missing values.

```
list if ownchild < .
```

Now list all the observations *with* missing values for variable OWNCHILD. (Hint use either the == or the => comparison operators.)

Another way to identify missing values (or non-missing values) is using the function **missing**. Notice the exclamation point (!) means *not*.

```
list if missing(ownchild)  
list if !missing(ownchild)
```

Now list the data if number of children is at least 2 *and* is not missing. Use the “and”

(ampersand &) to list two or more expressions in the **if** statement.

```
list if ownchild >=2 & !missing(ownchild)
```

So far, we've explored continuous variables using descriptive statistics. Let us develop descriptive statistics for categorical variables (those with distinct categories like SEX categories MALE or FEMALE, etc.).

Let's create frequency tables for our categorical variables.

Exercise 14:

How many people are reported in each sex category?

```
tabulate sex
```

Note, you can also use the shortened keyword **tab** instead of **tabulate**.
How many people are reported in each of the marital status categories?

Exercise 15:

Let's construct a cross tabulation between two categorical variables SEX and MARITAL.

```
tabulate sex marital
```

How can we see cell percents, row percents, column percents, and a Pearson's chi-square statistic to test the association between SEX and MARITAL?

```
tab marital sex, cell column row chi2
```

Analysis: Based on these data, we can not draw a conclusion about the association between sex and marital status.

How do you find *all* the possible options for the TABULATE command?
Recall that you can use **Help** from the top drop-down menu try ...

Help ► Search ... *type 'tabulate'*

What if you forget the command to run frequencies altogether? Yikes!
Use the point-and-click method: from the top drop-down menu try ...

Statistics ► Summaries, tables & tests ► Tables ► One-way tables ► *choose categorical variable and submit*

Labels

There is something unsatisfying with the marital status variable. Although we get the frequency category breakouts, we have to refer back to the data definition table to recall what the categories represent. We can fix this using a **label** statement. By way of contrast first we put a label on a single variable, and then put a series of labels on a variable's values.

Exercise 16:

Use the command **describe** to view your data again.

```
describe
```

Now put a descriptive label on the variable LNAME.

```
label variable lname "Last Name"
```

Use the command **describe** again to make the comparison.

Exercise 17:

Assigning a label to a value of a variable is a two-step process.

- 1) Create the value label first (**define**) and then
- 2) Assign the value label to the variable (**values**).

Put a descriptive label on the categorical values of the variable MARITAL. (Code this long **label** syntax on one line. We'll see how to remedy this awkward code shortly.)

```
label define marlab 1 "Married, Civilian Spouse Present" 2 "Married,  
Armed Forces Spouse Present" 3 "Married, Spouse Absent (exc.  
Separated)" 4 "Widowed" 5 "Divorced" 6 "Separated" 7 "Never Married"
```

```
label values marital marlab
```

Now retry the tabulation of the MARITAL variable. Also see what happens using the **describe** command.

What if you want to see the original values again without the labels? Use the option **nolabel** as follows.

```
tab marital, nolabel
```

How do we fix a piece of awkward long syntax (like the above long label command)? Stata has no default end-of-command symbol but rather uses "Enter" to mark the end of a command. But if you "Enter" before you're finished, you may not submit what you intended.

There are at least two ways to let STATA know that the end of your line is not the end of the command, that you want the command to run onto the next line. The first way is using a comment (like `*/ {code} /*`), and another way is using a `#delimit` (“pound-delimit”) command. We’ll look at both ways.

Exercise 18:

A) Tell STATA that we want our commands to continue onto the next line by using comments.

```
label define marlab /*
*/ 1 "Married, Civilian Spouse Present" /*
*/ 2 "Married, Armed Forces Spouse Present" /*
*/ 3 "Married, Spouse Absent (exc. Separated)" /*
*/ 4 "Widowed" /*
*/ 5 "Divorced" /*
*/ 6 "Separated" /*
*/ 7 "Never Married", modify
```

Note that **modify** was added as an option at the end of the **label** command. This is necessary because STATA by default will not alter a label that has already been created. To indicate that you intend to overwrite a label include the option **modify**. Alternatively, you could have dropped the label beforehand and then defined it again. This is how you drop a label.

```
label drop marlab
```

You can use another comment symbol as well, the one that says “ignore the rest of the line *and* ignore the carriage return”.

```
label define marlab ///
 1 "Married, Civilian Spouse Present" ///
 2 "Married, Armed Forces Spouse Present" ///
 3 "Married, Spouse Absent (exc. Separated)" ///
 4 "Widowed" ///
 5 "Divorced" ///
 6 "Separated" ///
 7 "Never Married", modify
```

B) Tell STATA that we want our commands to continue onto the next line by using the command `#delimit`. For example, we’d like the new delimiter (the symbol that tells STATA that a command is concluded) to be a semi-colon (;).

```
#delimit ;
```

```
label define marlab
 1 "Married, Civilian Spouse Present"
 2 "Married, Armed Forces Spouse Present"
 3 "Married, Spouse Absent (exc. Separated)"
 4 "Widowed"
 5 "Divorced"
 6 "Separated"
```

```
7 "Never Married", modify;
```

Note: the **#delimit** does not apply to syntax typed into the Command window; those are always executed when you press the Enter or Return. The **#delimit** command applies only to do-files and ado-files

Dropping Variables or Observations

Next, we will remove a variable from our dataset (i.e. remove a “column”). We will then subset this data retaining particular observations (i.e. remove a “row”).

Exercise 19:

First, view the number of missing values for the variable OWNCHILD by using the **missing** option.

```
tab ownchild, missing
```

Since there are quite a number of missing values, we are going to drop this variable from the dataset.

```
drop ownchild
```

You can also drop individual observations using the **drop** command. Suppose your interest is in people ages 18 to 27, more or less the college and early career ages.

```
drop if age > 27
```

Now **list** the data again to see the result.

You’ve just done a lot of work on your data. Once you have your data the way you like it, it’s best to save it so you don’t have to reconstruct it again.

Exercise 20:

Save the data currently in memory. From the top drop-down menu, click on the following.

File ► Save As ► *navigate to your folder, name it something descriptive, click Save*

Note: if you ‘Save’ the data rather than ‘Save As’, STATA will save the data to the same file as was originally imported into STATA, and thus overwrite the data file in your folder with the one in memory.

Creating New Variables

First, import a new, larger dataset into STATA called morg05s2.dta, still based on the 2005 MORG data.

Exercise 21:

Open a new dataset MORG05s2.dta using one of the methods discussed in the early

exercise. (Hint: File ► Open ..., or find the command in the Review window that starts with **use** ..., or scroll up in your do-file Editor to find the command that starts with **use**...) Look at the new data with **describe**, **inspect**, **codebook** and/or **list** commands.

One awkward variable is the earnings per hour (EARNHRE) in the unit of pennies. Let's convert this to dollars.

Exercise 22:

Create a new variable EARNHRED that represents earnings per hour in unit dollars.

```
generate earnhred = earnhre / 100
```

Note: you can use the keyword **gen** instead of **generate**.

Use the command **summarize** to compare EARNHRE with EARNHRED as follows.

```
summarize earnhre earnhred
```

Suppose you want to change a variable that's already been created. Use the **replace** command.

Exercise 23:

If the number of years of college is missing, we are going to assume that there are zero years of college. First show the frequency of the years of college, including missing values.

```
tab yrroll, missing
```

Next, replace the missing values with the value zero.

```
replace yrroll = 0 if yrroll == .
```

Run the tabulation again to compare the results to the original.

We will compare two different ways to create a new categorical variable, first with **replace** and then with **recode**.

Exercise 24:

Create a new indicator (dummy or (0,1)) variable from the variable VETERAN. If VETERAN takes on values 1, 3, or 5, the dummy variable is 1 (yes, a veteran), and if VETERAN is equal to 6, then the dummy variable is 0 (no, not a veteran).

First look at the variable VETERAN using the TABULATE command, with value labels and then without value labels.

```
tab veteran, missing
```

```
tab veteran, missing nolabel
```

Now create a new variable that will become our indicator and to start, set it to missing.

```
gen vetcat1 = .
```

What does VETCAT1 look like? Just to get a sample, show the first ten observations of variables VETERAN and VETCAT1.

```
list veteran vetcat1 in 1/10
```

Next **replace** the values of VETCAT1 according to the categories described above.

```
replace vetcat1 = 1 if (veteran == 1 | veteran == 3 | veteran == 5)  
replace vetcat1 = 0 if (veteran == 6)
```

Note the “or” symbol | . Also, note the parenthesis () are optional but may make reading the conditional statement easier.

Check VETERAN against VETCAT1 with a tabulate as follows.

```
tab veteran vetcat1
```

Now try this again using **recode**. Create a new variable and set it to equal VETERAN.

```
gen vetcat2 = veteran
```

Next, recode the variable VETCAT2 with 1 through 5 assigned to equal 1 (this will work in this case since no individual in our sub-sample was assigned a value of 2 or a value 4 for VETERAN), and 6 assigned to equal zero.

```
recode vetcat2 1/5=1 6=0
```

Now compare VETCAT1 with VETCAT2 with a tabulate.

Exercise 25:

Use **recode** to create a new dummy variable called NATIVE (native born versus foreign born) from the variable citizenship status where values for PRCITSHIP between 1 and 3 are assigned a 1 for the new variable NATIVE, and values 4 and 5 are assigned a zero. Be sure to check it.

Regression

Now let’s develop an ordinary least squares regression model. Before we do, let’s open a LOG file, that is a file that saves everything that comes out of in the Results window.

Exercise 26A:

Open a log file so that you can save your output.

Method 1) Navigate via the drop-down menu.

File ► Log ► *navigate to your folder, name it something descriptive, click Save*

Method 2) Type in the Command window or in your do-file:

```
log using "C:\ {your path} \morg05s1.dta", replace
```

Now we're ready to analyze these data.

Exercise 26:

Run a regression with earnings per hour as the dependent variable and age as the independent variable.

```
regress earnhre age
```

Analysis: In the preliminary analysis, it appears that age is a positive predictor for earnings per hour.

Let's add a categorical independent variable to the model. First we need to generate dummy variables for the categories.

Exercise 27:

To add SEX to the model, first create dummy variables as follows.

```
tabulate sex, gen(sx)
```

The option **gen** (generate) creates dummy variables for each level of SEX. The 'sx' designation tells STATA what the new dummy variable will be called. In this case SX1 and SX2 for the two categories of SEX.

Check both the dummy variables by doing tabulates of the following form.

```
tabulate sex sx1  
tabulate sex sx2
```

SX1 and SX2 aren't very descriptive. Let's rename the variables for ease of interpretation. Just look at the crosstab above to confirm their definitions.

```
rename sx1 male  
rename sx2 female
```

Next run the regression with the additional dummy variable for FEMALE (with reference category MALE).

```
regress earnhre age female
```

Analysis: Based on these data and controlling for age, the model suggests that being

female is associated with less earnings per hour.

Now generate a new set of dummy variables for all levels of the RACE variable. Note that by *not* renaming the RACE dummy variables, you can add them to the model in the form of RACE2-RACE7, that is, with just a dash between the like-named variables with names that end in sequential integers. Add the RACE variable indicators to your model and run it.

What's your reference category for RACE?
How do you interpret your model?

Save your dataset

```
save "H:\Courses\Teaching\STATA\0702\morg05s2_modified.dta", replace
```

Let's try an analysis of variance.

Exercise 28:

Compare the differences in earnings per week between males and females using a one-way ANOVA model.

```
oneway earnwke sex
```

Also compute the mean of the males and females by adding a **tabulate** option.

```
oneway earnwke sex, tabulate
```

To include the covariates age and highest grade completed in the analysis, use the **anova** command as follows.

```
anova earnwke sex age ihigrdc, continuous(age ihigrdc)
```

What's your interpretation of this model?

Graphics

Let's begin with a histogram of weekly wages.

Exercise 29:

Create a histogram of earnings per week.

```
histogram earnwke
```

Try some box plots.

Exercise 30:

Create a boxplot of earnings per week.

```
graph box earnwke
```

Re-create the box plot grouping by sex.

```
graph box earnwke, by(sex)
```

Do another box plot of earnings per week grouped by NATIVE.

Now try some scatter plots.

Exercise 31:

Create a scatter plot of hours worked last week against earnings per week.

```
graph twoway scatter hourslw earnwke
```

Note: the alternative short-hand for a scatter plot is as follows.

```
twoway scatter hourslw earnwke
```

Try that again only with age versus earnings per week.

Add a regression line to a scatter plot.

Exercise 32:

Create a regression line to represent the relationship between hours worked last week and earnings per week.

```
twoway lfit hourslw earnwke
```

Next, we'd like both the scatter plot *and* the regression line together on the same plot as follows.

```
twoway (scatter hourslw earnwke) (lfit hourslw earnwke)
```

Note: the parenthesis () are not optional here.

Now add a label to the points in the scatter plot. Add labels for SEX.

```
twoway (scatter hourslw earnwke, mlabel(sex)) (lfit hourslw earnwke)
```

Next, separate the graph by sex and add a confidence band along with the predicted value using option **lfitci**.

```
twoway (scatter hourslw earnwke) (lfitci hourslw earnwke), by(sex)
```

Exercise 33:

Create a matrix of scatter plots for several variables at once.

```
graph matrix hourslw earnwke ihigrdc
```

How can you find *all* the graph command options?

If you completely “forget” how to graph in Stata, not even remembering the keyword **graph**, what can you do?

Merging Datasets

Often researchers will need to combine two or more datasets. Usually there are one or more key variables, a set of variables that uniquely identifies each record that will be used to match datasets together. In our current dataset, just one variable identifies each record uniquely. That is HHID.

Exercise 34:

Suppose you’d like to explore more about why certain individuals did not work a 40 hours in the previous week. You’d like to include one more variable in the dataset, WHY3594 that describes a reason. Merge datasets MORG05S2 and MORG05S2_PARTTIME_REASON on the key variable HHID.

Three steps:

1) Sort the dataset in memory by the key variable(s) and save dataset.

```
sort hhid
```

```
save "H:\Courses\Teaching\STATA\0702\morg05s2_modified.dta", replace
```

2) Open the second dataset, look at it, and sort by same key variable(s).

```
use "H:\Courses\Teaching\STATA\0702\morg05s2_parttime_reason.dta",  
clear
```

```
describe
```

```
sort hhid
```

3) Merge the two datasets together, the one in memory and the one saved to a file, on the key variable(s).

```
merge hhid using  
"H:\Courses\Teaching\STATA\0702\morg05s2_modified.dta", unique
```

Look at the newly added variable WHY3594.

```
list hhid hourslw why3594
```

Note: not only have the datasets been combined, but the variable `_merge` has been added to your dataset. `_merge` will “mark the source of the resulting observation”.

Could individuals not be working a full 40 hours because they are students? Now merge the current dataset in memory to the dataset MORG05S2_STUDENT, again using the key variable HHID. The new data provides an extra variable STUDFTPT that tells if an individual is a student, either full-time or part-time. Remember, you already have a variable called `_merge` in your dataset. You will need to drop it before you can merge again because the merge will automatically try to create it and so get an error.

```
drop _merge
```

What does the variable `_merge` look like after combing these final two datasets?

Reshape Data

Sometimes you need to re-orient data to make it compatible with your analysis method. In STATA, the terms used to describe data orientation are 'wide' and 'long'.

Wide data, like it sounds, extends across many columns (variables) of the dataset. Long data, also like it sounds, extends down many rows (observations) of the dataset.

For example, here is a small STATA dataset called MORG_EARNWKE with three variables: HHID, YEAR, and EARNWKE (earnings per week).

| hhid | year | earnwke |
|-----------------|------|---------|
| 57910760998880 | 2005 | 250 |
| 084313380994691 | 2005 | 170 |
| 100969066154434 | 2005 | 420 |
| 57910760998880 | 2004 | 250 |
| 100969066154434 | 2004 | 400 |
| 57910760998880 | 2003 | 300 |
| 084313380994691 | 2003 | 120 |
| 100969066154434 | 2003 | 360 |

Is this a wide dataset or a long dataset? Here's a clue. When you want to add more information to this dataset, does it go in as more rows or more columns? Rows (it gets longer), so it's a LONG dataset. Let's look at the pieces of the **reshape** command.

```
reshape wide earnwke, j(year) i(hhid)
```

wide tells reshape that we want to go from long to wide

earnwke tells Stata that the variable to be converted from long to wide is **earnwke**

j(year) tells reshape that the suffix of **earnwke** should be taken from the variable **year**

i(hhid) tells reshape that **hhid** uniquely identifies observations in the wide form

Exercise 35:

Open and then re-orient the dataset MORG_EARNWKE from 'long' to 'wide' using the **reshape** command. Then look at it with the command **list**.

```
use "C:\ {your path} \morg_earnwke.dta", clear
```

```
reshape wide earnwke, j(year) i(hhid)  
list
```

Let's save this dataset and then try to get back to the long orientation.

```
save "H:\Courses\Teaching\STATA\0702\morg_earnwke_wide.dta", replace  
reshape long earnwke, j(year) i(hhid)
```