RAL

API-210i
Empirical Methods II
Spring 2003

A SHORT INTRODUCTION TO STATA

Course materials and data sets will assume that you are using Stata to complete the analysis. Stata is available on all of the computers in the Kennedy School's computer lab. However, the lab is in high demand, so if you have a home computer you may want to purchase a copy of Stata from the CMO. Stata is available for Windows95, WindowsNT, Macintosh, and UNIX operating systems. The Stata User's Guide is also available from the CMO.

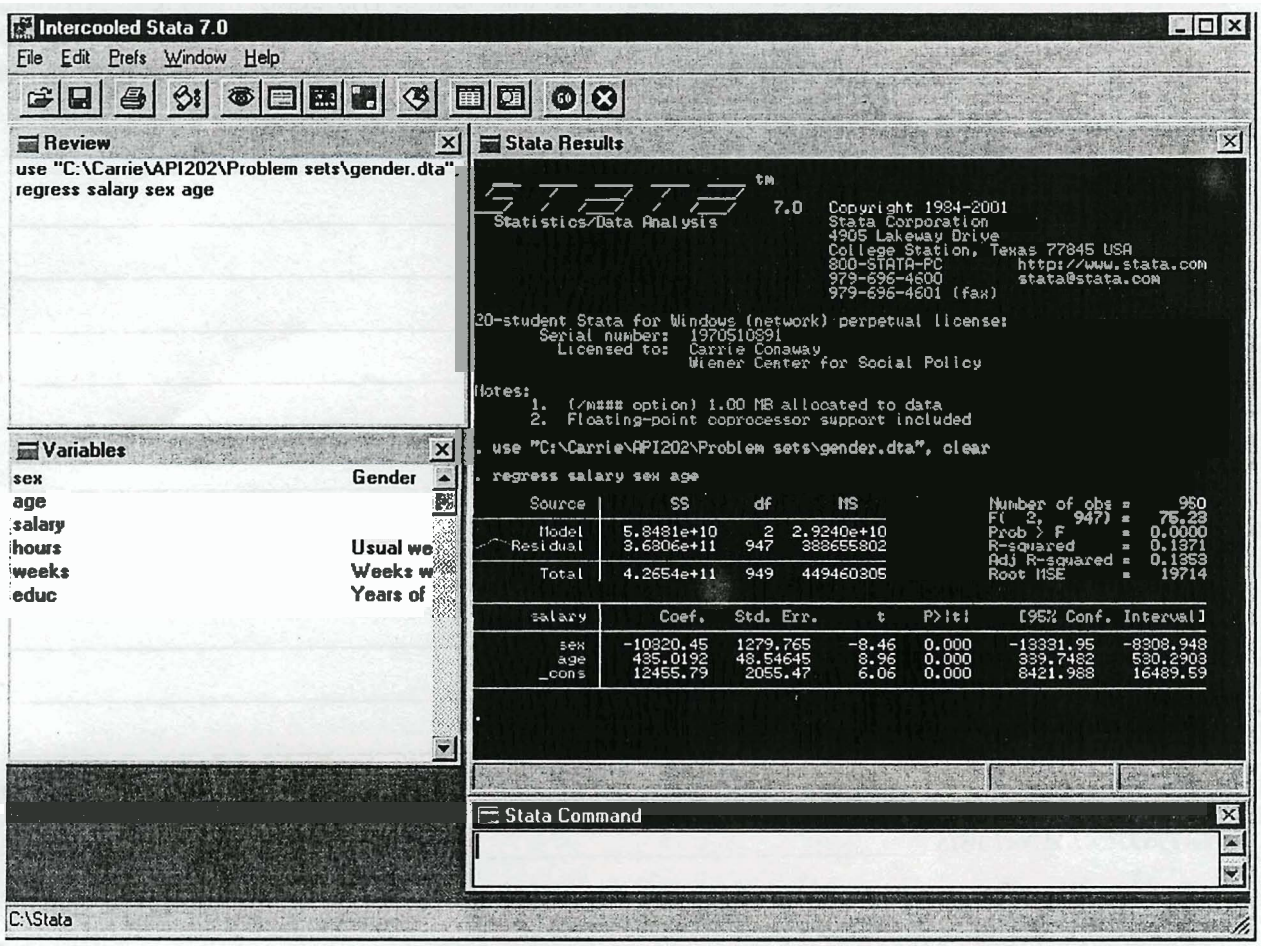
The commands outlined below assume that you are using Stata Release 7.0 for Windows. However, most will also work with Stata Release 6.0 and with the Macintosh and UNIX versions of the software. And we assume all of them will work with the newest Stata Release 8, which will be installed on all the computers at the computer lab this spring. Throughout this text, anything appearing in `Courier` font is a Stata command, whereas anything in Arial font is a comment or hint. Menu commands are indicated as, e.g., `File | Open`, to indicate that you first go to the `File` menu and then choose the `Open` option.

GETTING STARTED

1. How Does Stata Work?

Stata is a powerful tool for conducting statistical analyses. It can do everything you did last semester with Excel and much more. It is different from many other software programs you may be familiar with, however, in that it operates primarily by manually entering or programming commands rather than by choosing options from drop-down menus. You can work with Stata interactively by entering commands individually, or you can write programs (called "do files") and have Stata run a series of commands in a row.

Here is what a typical session in Stata looks like.



Stata's "Command Central", the Command window, is at the bottom center of the screen. When working with Stata interactively, all commands are entered here. What is a command? It is a statement that tells Stata what to do – i.e., to open a file, to run a regression, to calculate a mean of a variable, etc.

The Review window, in the upper left hand corner, shows a list of all the commands you have already run. (Here, it shows that I have opened a data file and then run a regression.) If you click on a previously-run command in the Review window, it will appear in the Command

window and you can edit it or run it again. The Variables window shows a list of all the variables in the data set you have open, and finally the Results window shows you the results of your most recent statistical analyses. (Here, it shows the results of an ordinary least squares regression of salary on age and sex.)

Whenever you start Stata, you will first need to open the data file you wish to work with and open a log file (which saves your analysis in a file that you can print or edit later). Then you enter commands sequentially to create variables, perform statistical analysis, and so on. The next four sections of this guide provide a list of the basic commands for getting started with Stata as well as some hints on how to use them.

2. Managing the Workspace

use

You will typically download data from the course web site to use in completing your homework. Save the data file (which will end with ".dta") someplace convenient – say, in a folder on your M drive called "API-210i". To open the file in Stata, enter the following into the Command window:

```
use "m:\api-210i\gender.dta"
```

(If you're using Stata interactively, you can also choose `File | Open` and select the file you want from the directory.)

If there are unsaved changes to any data in the workspace which you want STATA to overwrite before loading the new data, you need to clear Stata's memory as follows:

```
use "m:\api-201\gender.dta", clear
```

save

If you have made changes to the data file and wish to permanently save them, the command is:

```
save "m:\api-210i\gender.dta"
```

If you want Stata to overwrite an existing data file called "gender.dta":

```
save "m:\api-210i\gender.dta", replace
```

exit

To exit Stata, simply type:

```
exit
```

However, if there are unsaved changes to the data in the workspace (which is often the case), you must either save the changes or explicitly abandon them before you can exit. To explicitly abandon changes to the data and exit:

```
exit, clear
```

3. Logging a Stata Session

log

To save ("log") your results, you will need to create a log file. Stata gives you two choices of file formats for your log file, .log (text file) and .smcl (formatted log file). The .smcl files will look nicer when printed, but if you want to cut and paste your output into your homework you may prefer working with the .log files instead.

To start a log file interactively, choose `File | Log | Begin`, select the directory you want to save the log file in, and give it a name (such as `assign1`). Alternately, you can click on the fourth icon from the left on the icon bar, which looks like a scroll.

To start a log file via the Command window, first enter:
`set logtype text` (for text output)
or
`set logtype smcl` (for formatted output – this is the default)

Then create the log file with the following command:
`log using "m:\api-210i\assign1.log"` (for text output)
or
`log using "m:\api-210i\assign1.smcl"` (for formatted output)

If you've already created a log file for the particular project you're working on and you try to open that same log file again interactively, Stata will ask you if you want to view the old log file, replace (i.e. overwrite) it, or append to it. If you're working via the Command window and you know you want to overwrite your old log, you can tell Stata this directly using the command:
`log using "m:\api-210i\assign1.log", replace`

If you want to close a log file (for example, if you wish to begin a new one), choose `File | Log | Close`, or click on the scroll icon and choose close.

4. Viewing the Data

Once you have opened a data set with the "use" command, you may wish to look at the variables and observations in spreadsheet format. Stata provides two ways to do this, "browse" and "edit". The browse command lets you see the data but not make changes, whereas the edit command allows you both to browse and to make changes. It is probably best to use browse unless you actually intend to make changes to your data manually; otherwise you may accidentally change something and ruin your data.

To browse, enter `browse` into the Command window or select the Browse icon (third from the right, a spreadsheet with a magnifying glass on it). To edit, enter `edit` into the Command window or select the Edit icon (fourth from the right, a spreadsheet with no magnifying glass).

5. Creating and Submitting a Do File

Although Stata can be run interactively by just typing one command at a time, Stata commands can also be submitted in batches by using a "do file." A do file is simply a text file which contains a series of Stata commands. You enter the Stata commands in the same order as you would enter them interactively, and Stata then runs these commands automatically instead of your having to type them in line by line.

For your problem sets, it is strongly recommended that you use do files. Some of the problem sets will require many Stata commands, and it is inevitable that you will need to make changes and run these series of commands a number of times. When you have all of your commands in a single file, it is much easier to go back to that file and make the necessary changes than to have to retype every command.

Creating a Do file

To start creating a do file, click on the Do file editor button (fifth from the right, looks like an envelope with a pencil on it), choose the Do file editor option under the Window menu, or type `doedit` in the Command window. Note that since a do file is a written list of commands as entered in the Command window, you cannot use the Stata menus within a do file. Instead you need to use the typed (Command window) commands.

As discussed above, you will want to have a log of your results. As before, you will first want to choose which type of log file you want by entering into your do file either:

```
set logtype text        (for text output)
```
or
```
set logtype smcl        (for formatted output).
```

Then create the log file with the statement:

```
log using "m:\api-210i\assign1.log"     (for text output)
```
or
```
log using "m:\api-210i\assign1.smcl"    (for formatted output)
```

If you have already created a log file called "assign1.log" and wish to replace (overwrite) the old one, use the following command:

```
log using "m:\api-210i\assign1.log", replace
```

Next, you will want to have Stata open the data file you wish to work with. The command is:

```
use "m:\api-210i\gender.dta"
```

If you already have a data set open and want to open a new one, use the following command:

```
use "m:\api-210i\gender.dta", clear
```

You may also wish to include some of the following commands at the beginning of your do file:

`clear`	clears any open data sets out of Stata's memory
`set more off`	keeps you from having to hit a key every type your output runs over a single page length
`#delimit ;`	tells Stata that every line ends with a semicolon rather than a "Return" or "Enter" key. Convenient if you are running analyses with enough variables that keeping them all on one line means some scroll off the edge of the page in the Do file Editor.

You can add comments to your file by beginning the line you wish to comment out with an asterisk. This helps organize your file by allowing you to include sentences and phrases that aren't in Stata syntax. For example, you could label major sections or remind yourself of how you created a variable.

So, the beginning of a typical do file might look like this:

```
* Assignment 1
clear
set more off
set logtype text
log "m:\api-210i\assign1.log"
use "m:\api-210i\gender.dta"
```

After this, you just enter commands in the same order and with the same syntax as you would have if you had been performing your analyses interactively in the Command window. (See

below for examples of the most frequently used commands.) When you're done, save the do file by choosing `File | Save` from within the Do File Editor window. By convention, do files usually have a `.do` extension, and it is best to keep this extension on your do files as well.

Submitting a Do file

When you are ready to run your do file, select `File | Do` from the main Stata menu and double-click on the name of your do file. Alternately, in the Do File Editor window, you can click on the "Do Current File" icon (second from the right, with a downward pointing arrow and lines on a page of paper). Make sure that you have saved any changes you have made to your do file before you run it.

If there are any errors in your do file, you will get an error message and Stata will stop running the do file. The error message will indicate the line where the syntax needs to be changed. Open up the do file, make the necessary changes, and make sure to save the revised version of your do file before resubmitting it in Stata. If you want to interrupt a do file that is executing, just press <Break>.

WORKING WITH DATA

1. Using Operators

Stata uses the following arithmetic operators:

+	add
-	subtract
*	multiply
/	divide
^	raise to the power

For relations, Stata uses:

==	equal
~=	not equal
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to

Note that a single equal sign (=) is used when assigning a value to a variable:

```
gen wage = salary/(hours*weeks)
```

but a double equal sign (==) is used when asking Stata to make a comparison:

```
replace fulltime = 1 if hours == 40
```

For logical operations, Stata uses:

&	and
|	or (pipe sign; what you get when you hit Shift and the "\" key)
~	not

2. Creating and Changing Values of Variables

generate
Creates a new variable, in this case yearly hours "yrhrs", defined as "hours" multiplied by 50:

```
generate yrhrs = hours * 50
```

You can abbreviate this command with "gen". Note that Stata will tell you if any missing values were generated by attempting to perform a calculation with missing information. For example, if one of the observations was missing information on "hours," Stata would set "yrhrs" equal to missing for this observation. (See further notes about missing data later in this section.)

replace
Changes the value of an existing variable:

```
replace yrhrs = hours * 52
```

You can change the value of an existing variable for only certain observations, in this case where "sex" is 1:

```
replace yrhrs = hours * 52 if sex == 1
```

(Notice the use of the single and double equals signs.)

Once a variable with a particular name has been generated you can't generate another with the same name. This is true even if you only generated valid (non-missing) values for part of the data set, as in the second example above. Instead, you must replace the old one. For example, suppose you want to calculate hours worked only for people who are employed (coded with value 1 in the variable employed). For unemployed people and people who aren't in the labor force (coded as 0 in the variable employed), you want their hours worked to equal zero. Here's how to do it:

This code won't work:

```
gen yrhrs=hours*52 if employed==1
gen yrhrs=0 if employed==0
```

But this code will:

```
gen yrhrs=hours*52 if employed==1
replace yrhrs=0 if employed==0
```

recode
Makes multiple changes to the value of an existing variable. For example, suppose you have a variable called "age" that contains the age of each person in your study. But you want the ages to be collapsed into groups, say age less than 24, age 25 to 34, age 35 to 44, and so on. To do this, create a new variable called "agegroup" and recode it as follows:

```
gen agegroup=age
recode agegroup min/24=1 25/34=2 35/44=3 45/54=4 55/64=5 65/max=6
```

When recoding your data, be careful not to overwrite your original variable (in this case, by simply recoding age rather than generating a new variable and recoding the new one) unless you are absolutely sure you will never want to use the original codes again.

An important note about missing data
Sometimes you will not have valid data for every observation in a particular variable. For example, a survey respondent may have skipped a particular question, or the question may not be asked of certain types of respondents. Observations without valid data are entered in Stata with a "." and are considered missing. Any time you perform a calculation (adding, multiplying, etc.) on a variable with missing data, the resulting new data will be missing for any observations in the original variable that had missing data – which is what you want. But you have to be

careful when recoding or replacing values of variables with missing data, since Stata considers missing data to be larger than any other valid observations in the data set.

Here's why. Suppose you are recoding a variable for education, called "educ", into a new variable, "hsgrad", which has two categories (high school graduate and less than high school). Everyone who has a value of 12 or more on "educ" should get a 1 in "hsgrad", and everyone with a value of 11 or less should get a 0 in "hsgrad". Now suppose that some people have missing data on the educ command. If you create the new "hsgrad" variable by simply telling it to code everyone who has more than 12 years of education as a 1, anyone with missing data on "educ" will also get a 1 – and this is probably not your intention. Instead, you need to tell Stata to generate the new variable, "hsgrad," only for valid observations of the old variable, "educ".

This program will code missing data in educ as hsgrad=1:

```
gen hsgrad=0 if educ<12
replace hsgrad=1 if educ>=12
```

This program will code anyone with missing data for educ as missing for hsgrad, as well:

```
gen hsgrad=0 if educ<12
replace hsgrad=1 if educ>=12 & educ~=.
```

A shortcut for creating dummy variables

You will find that you will often want to create "dummy" variables – variables that take on a value of either zero or one. The hsgrad variable we just created is an example of a dummy variable; it equals 0 if the person did not graduate from high school and 1 if s/he did. We originally created this variable in a two-step process, first generating the variable and then replacing some of its values. But this can actually be done in one statement, as follows:

```
gen hsgrad=(educ>=12) if educ~=.
```

This tells Stata to create a variable that equals 1 when whatever is in the parentheses is true (in this case, when the variable educ is greater than or equal to 12) and 0 when it is not true. The if statement at the end tells Stata not to generate any values for this variable if educ is missing, which avoids the problems of recoding with missing data as discussed above.

3. Using Functions

Functions are special calculations used with other commands, such as `generate` or `replace`. Stata has the capability to calculate many functions. Here are some examples of the most commonly used ones.

ln(x)

Calculates the natural log of x, where x may be a constant or a variable such as "salary", or an equation such as (wage + tip)

`ln(1.5)` or `ln(salary)` or `ln(wage + tip)`

In a command, you might use the ln function like this:

```
gen logsal = ln(salary)
```

sqrt(x)

Calculates the square root of x, where x may be a constant or a variable such as "salary", or an equation such as (wage + tip)

`sqrt(1.5)` or `sqrt(salary)` or `sqrt(wage + tip)`

In a command, you might use the sqrt function like this:
```
gen utility = sqrt(wealth)
```

norm(Z)
Tells you the probability associated with the particular Z value on the normal density function. For example, norm(1.96) = 0.975, since 97.5% of the data in a normal distribution lie to the left of Z=+1.96.

4. Deleting Variables and Observations

drop
The drop command can delete either variables or observations. Deleting a variable removes an entire variable (column) from the data set, whereas deleting an observation removes an entire observation (row) from the data set. Be careful when doing this – the variables and observations are permanently deleted once you save the data file!

To eliminate a variable, in this case "hours":
```
drop hours
```

More than one variable can be deleted with a single statement, in this case "yrhrs" and "hours":
```
drop hours yrhrs
```

To eliminate observations, in this case those for which "fulltime" is one:
```
drop if fulltime == 1
```

5. Describing and Examining Data

summarize
Calculates descriptive statistics on all variables in the workspace:
`summarize` (can be abbreviated as "sum")

You can also summarize only some of the variables, in this case "salary" and "hours":
```
sum salary hours
```

You can see additional summary statistics, such as medians, if you use the "detail" option:
```
sum salary, detail
```

You may want only to summarize certain observations, in this case when "agegroup" is less than or equal to 2:
```
sum salary if agegroup <= 2, detail
```

You may wish to summarize a variable, in this case "salary", separately for each value of another variable, in this case "agegroup":
```
sort agegroup
by agegroup: sum salary, detail
```

tabulate

Makes a table of each possible value of a variable and the number of observations in the data set which have that value. Note that you can only "tabulate" variables which take on fewer than 99 different values.

```
tabulate agegroup          (can be abbreviated "tab")
```

You can limit the calculation to certain observations, in this case those where "agegroup" is less than 3:

```
tab agegroup if agegroup < 3
```

You can calculate means of another variable, in this case "salary", for each value of the variable being tabulated, in this case "agegroup". This will tell you the average salary in each of the categories of the variable "agegroup".

```
tab agegroup, sum(salary)
```

You can produce cross-tabulations by specifying two variables. The first variable listed goes in the rows, the second in the columns.

```
tab fulltime agegroup
```

To calculate the percentage of observations in each cell:

```
tab fulltime agegroup, cell
```

You can calculate the percentage of observations in each column and/or row, as well, by adding any combination of "cell", "col", and/or "row" after the comma, e.g.:

```
tab fulltime agegroup, row col
```

list

Prints all variables and observations to the screen. You'll probably never want to do this since your data sets will be too large.

```
list
```

You can print a limited set of variables, in this case "fulltime" and "salary":

```
list fulltime salary
```

You can print a limited set of observations, in this case observations 15 through 26:

```
list fulltime salary in 15/26
```

You could also print a limited set of observations according to another criteria, in this case "agegroup" being equal to 2:

```
list if agegroup == 2
```

codebook

Provides even more information (mean, standard deviation, range, percentiles, labels, number of missing values, etc.) about a variable:

```
codebook salary
```

6. Labeling Variables and Values

Labeling variables and values helps you keep track of how you coded your variables and what they represent. It takes just a couple of seconds to add labels, and it can save you lots of time later when you can't remember what the a code of "4" means in your education group variable, for example, or how the variable "wages2" differs from "wages1".

label variable
To attach a label to a variable, in this case the label "Annual Salary" to the variable "salary":
```
label variable salary "Annual Salary"
```

label value
To label the values of a variable, you first need to define a value label to tell Stata which label goes with which value. The example below creates a value label, abbreviated "ptft", which when applied to a variable will label any values of 0 as "Part time" and any values of 1 as "Full time". You can apply the same value label to more than one variable.
```
label define ptft 0 "Part time" 1 "Full time"
```

Next, you need to apply the value label to the particular variable whose values you want to label, in this case "fulltime":
```
label value fulltime ptft
```

If you want to change a label after you have defined it, you can use the "modify" option:
```
label define ptft 0 "PT" 1 "FT", modify
label value fulltime ptft
```
(Note that you must repeat the `label value` command in order for the changes to take.)

STATISTICAL ANALYSIS

1. Testing Hypotheses About Means

ttest
tests a hypothesis about a mean, in this case that the mean of the variable "wage" is 6:
```
ttest wage = 6
```

To test a hypothesis that the means of two groups are equal, in this case where the two groups are defined by a variable "sex":
```
ttest wage, by(sex)
```

To perform this test without imposing the restriction that the variance be equal in both groups:
```
ttest wage, by(sex) unequal
```

2. Estimating Linear Models (OLS and 2-Stage Least Squares)

regress
Calculates an ordinary least squares (OLS) regression, in this case for a regression of the dependent "salary" on the independents "age" and "sex". Note that the dependent variable is the first variable listed.
```
regress salary age sex
```

If you wish to only include observations with "age" equal to 30 in the regression:
```
regress salary age sex if age == 30
```

To run a regression with robust standard errors:
```
regress salary age sex, robust
```

To run two-stage least squares where "hours" is endogenous and "z1" is an exogenous instrumental variable:
```
regress salary age sex hours (age sex z1)
```

Running a regression automatically saves the statistics produced by the command in a temporary form called a scalar until the next regression is run. In the above example, the scalar "_b[age]" contains the coefficient estimate for the "age" variable while the scalar "_b[sex]" contains the estimate for the "sex" variable. The sum of squared residuals, which is sometimes useful in constructing tests of joint hypotheses, is saved temporarily as "_result(4)". These scalars can be useful in calculating predicted values and other statistics.

Note: If you run a regression containing more than 40 variables, Stata will return an error code saying:
```
matsize too small
```

To overcome this problem, reset the maximum number of variables Stata will estimate using the `matsize` command; the number should be greater than or equal to the total number of variables in the regression.
```
set matsize 150
```

predict
Calculates the predicted value for each observation using the coefficients from the last regression estimated and saves these as a variable called "yhat":
```
predict yhat
```

To calculate the residual for each observation using the most recently estimated regression model and save these as a variable called "ehat":
```
predict ehat, residual
```

test
Calculates an F-test of a joint hypothesis concerning the coefficients in the most recently estimated linear regression model, in this case with the null hypothesis H0: $\beta_{age} = \beta_{sex} = 0$:
```
test age sex
```

If you wish to construct an F-test that the two coefficients are equal to each other, but not necessary equal to zero:
```
test sex = age
```
(This particular example doesn't make much sense, since there's no reason to believe the effect of sex would be equal to the effect of age. A scenario where this kind of test is more logical might be testing whether the effect of race on wages is the same whether you are black or Hispanic. Then you might test whether the coefficient for a dummy variable for black equals the coefficient for a dummy variable for Hispanic.)

Note that if you run the "test" command after using a maximum likelihood technique such as logit or probit, Stata will produce a likelihood ratio chi-squared test rather than an F-test. While

they are calculated differently, the interpretation is the same – a significant F-test or chi-squared test means that the variables jointly contribute explanatory power to the model.

Estimating Non-Linear Models (Logit and Probit)

logit
Estimates a model suitable for a dichotomous dependent variable. In this case, the variable "fulltime" equals 1 if the individual worked full time all year and 0 otherwise. If you wish to estimate the probability of "fulltime" conditional upon "age" and "educ":
```
logit fulltime age educ
```

If you wish to find a predicted probability for each observation based on the most recent model run and save these as a variable called "phat":
```
predict phat
```

probit
Estimates a model suitable for a dichotomous dependent variable. In this case, the variable "fulltime" is one if the individual worked fulltime all year and zero otherwise. If you wish to estimate the probability of "fulltime" conditional upon "age" and "educ":
```
probit fulltime age educ
```

If you wish to find a predicted probability for each observation based on the most recent model run and save these as a variable called "phat":
```
predict phat
```

MISCELLANEOUS

1. Making Graphs

(This section is organized a bit differently from the other sections since all graphs are produced using the same basic command)

Histogram
This is the default when only one variable is specified, in this case salary:
```
graph salary
```

You can set the number of bars, in this case to 30:
```
graph salary, bin(30)
```

You can also draw a normal density over the histogram:
```
graph wage, bin(30) normal
```

To have STATA pick round values for the x- and y-axes:
```
graph salary, bin(30) xlabel ylabel
```

You can have STATA label specific values on the x- and y-axes:
```
graph salary, bin(30) xlabel(0,50000,100000) ylabel(0,0.25,0.5)
```

To have STATA graph only certain observations, in this case those for which "agegroup" is 2:

```
graph salary if agegroup == 2, bin(30) xlabel ylabel
```

To add a title:

```
graph wage, by(sex) bar mean ylabel title("Wages of Employed Persons,
By Gender")
```

To add one title on the top:

```
graph salary, bin(30) xlabel ylabel t1(First Title on Top)
```

To add two lines of title on the top:

```
graph salary, bin(30) xlabel ylabel t1(First Title on Top)
      t2(Second Title on Top)
```

The options b1 and b2 (bottom titles); l1 and l2 (left titles); r1 and r2 (right titles) work similarly, but on the bottom, left and right of the graph.

To add reference lines at x = 10000 and y = 0.1:

```
graph salary, bin(30) xlabel ylabel xline(10000) yline(0.1)
```

To label these lines at the top and at the right:

```
graph salary, bin(30) xlabel ylabel xline(10000) yline(0.1)
      tlabel(10000) rlabel(0.1)
```

After performing a regression, you may want to graph predicted and actual values of the dependent variable against the independent variable:

```
graph wagehat1 wage age, xlabel ylabel symbol(o.)
```

Scatterplot

This is the default if two variables are specified, in this case "salary" and "age":

```
graph salary age
```

Conditions, axes, titles, labeling and reference lines can be specified as above. For example:

```
graph salary age, xlabel ylabel t1(Salary by Age)
```

Boxplot

This is produced with a graph command followed by one variable, in this case "salary", and the "box" option:

```
graph salary, box
```

To not plot the outliers use:

```
graph salary, box s(i)
```

To produce parallel boxplots separated by the different values of a particular variable, in this case sex:

```
sort sex
graph salary, box by(sex)
```

To use box widths proportional to sample size:

```
sort sex
graph salary, box by(sex) vwidth
```

Conditions, y-axis options, most titles, and horizontal reference lines can be specified as described above with regard to histogram:

```
sort sex
```

```
graph salary, box by(sex) t1(Household Income by Agegroup) t2(Queens)
l1(Household Income) l2(thousands) ylabel yline(33000)
```

Bar graphs

This is produced with a graph command followed by one variable, in this case "salary", and the "bar" option. A second variable, in this case "agegroup" is used to define by groups. To produce a graph with bar heights representing the mean for each group:

```
sort agegroup
graph salary, bar means by(agegroup)
```

Conditions, y-axis options, most titles, and horizontal reference lines can be specified as described above with regard to histogram:

```
sort agegroup
graph salary, bar means by(agegroup) t1(Salary by Agegroup)
     t2(Title 2) l1(Mean Salary) l2(Another Title) ylabel
     yline(33000)
```

Printing your graph

Stata allows you to print (`File | Print Graph`) and save (`File | Save Graph`) your graphs. However, for the problem sets, the easiest way to incorporate your graph into a Word document is to copy the graph to the clipboard using `Edit | Copy Graph` and then paste it into your document.

2. Converting Data Files (Excel to Stata)

The easiest way to convert data files is to use the software program StatTransfer. This program is on the lab computers and allows you to convert your data **to** or **from** a variety of different file formats (Stata, SAS Transport, Excel, SPSS, QuatroPro, FoxPro, etc.).

To convert a file from Excel to Stata:

a) Click on the application StatTransfer in the "Data Analysis" folder.

b) Select "Excel Worksheet" for "Input File Type."

c) Use "Browse" to identify the Excel file you want to convert from. (If the first row of the worksheet contains the variable names, the program will use these as the variable names.)

d) Select "Stata Version 7" as the "Output File Type." (Since Stata 7.0 is a recent release, it is possible that the version of StatTransfer you're using will not have Stata Version 7 as an option. If this is the case, save it as a version 6 file; you should still be able to open the file in version 7.)

e) Type in the path and name of the file you wish to create.

f) Begin the conversion by clicking on "Begin Transfer."

Stata also allows you to read in binary and ASCII files directly. However, in most cases it is easier to first convert your data to a spreadsheet and then convert it to Stata using StatTransfer.

3. Using Matrices in Stata

Most commands and operations regarding matrices begin with the word 'matrix,' to tell Stata we are using a matrix, not a variable. Most operations follow the notation we've used all along.

1. CREATING A MATRIX

1A. ENTERING DATA INTO A MATRIX

We can enter data into a matrix by typing them in directly,

matrix A= (1,2 \ 3,4)

Stata uses commas to denote the number appears in the next column, and forward slashes ' \ ' to signify the next row.

Thus the matrix A above is the way to type in

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

1B. TO MAKE A MATRIX OUT OF VARIABLES IN YOUR DATASET

To create a matrix out of variables, we type

mkmat *var1 var2*, matrix(X)

This takes var1 and var2 as columns in a new matrix called X

1C. TO MAKE A DIAGONAL MATRIX OUT OF VARIABLES IN YOUR DATASET

To take a vector and place the elements of that vector along the diagonal of a matrix, type

matrix H=diag(*temp*)

Thjs takes the vector temp and creates a new matrix H with the elements of temp along the diagnol, and zeroes elsewhere.

2. DISPLAYING CONTENTS OF A MATRIX

We display the contents of a matrix using,

matrix list *matrixname*

To access a particular element of a matrix, we can index the matrix,

di VCV[2,2]

displays the element of matrix VCV in row 2, column 2. Note, it is always [row,column]

3. MATRIX OPERATIONS

3.A ADDITION/SUBTRACTION

To add one matrix to another, we simply use the '+' sign, and to subtract we use '-'

matrix C=A+B

creates a new matrix C as the matrix sum of A and B.

3B. MATRIX MULTIPLICATION

To multiply two matrices, we use ' *'

matrix C=A*B

Creates a new matrix C as the product of A and B. You must make sure the matrices are conformable, else you get an error from Stata.

3C. INVERSE OF A MATRIX

To take the inverse of a matrix, we type

matrix invX=inv(matrixname)

Stata will let you know if the matrix is non-invertible.

3D. TRANSPOSE

To get a matrix transpose, we use ' ' '