

THE UNIVERSITY OF MELBOURNE
Department of Economics

ECOM30003/90003: Applied Microeconomic Modelling

Introduction to Stata

This handout covers the basics of Stata: Stata's windows, using, managing and saving data, log files, do files and basic regression commands.

I. Stata

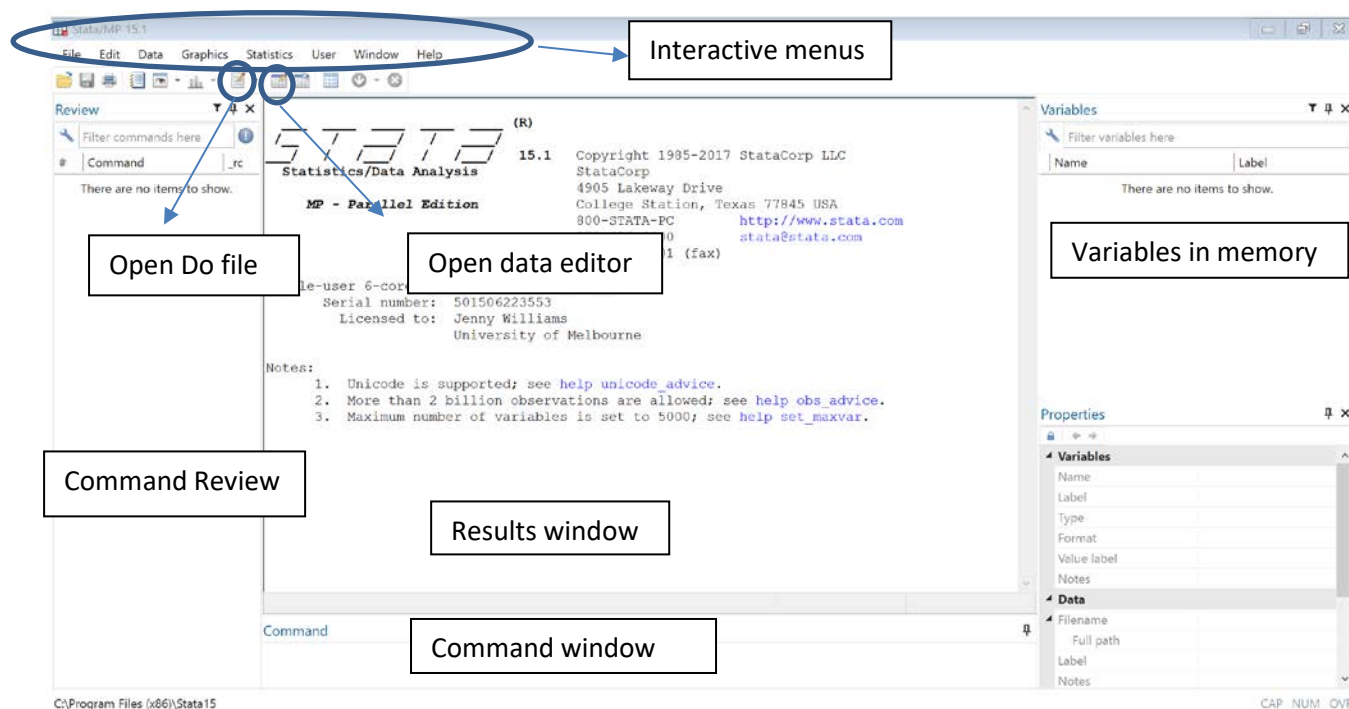
Stata is an integrated statistical analysis package. The official website is <http://www.stata.com/>. Its main strengths are handling and manipulating large data sets, and it has ever-growing capabilities for handling analysis of panel, cross-section and time-series data. It now also has pretty flexible graphics capabilities. The most recent version is *Stata* 16 and with each version there are improvements in computing speed, capabilities and functionality. It is also constantly being updated or advanced by users who write programs tailored to a specific need that are usually available on the web for you to integrate with your own software. The lab has *stata* 15.

II. Accessing Stata in the Labs.

To access Stata from a computer in the Lab go to Programs and double click on "Stata16". This will open Stata.

Students are also able to remotely access Stata using MyUniApps.

<https://studentit.unimelb.edu.au/myuniapps>



On top of the screen you have pull-down menus that allow you to edit, graph, and do basic statistics analysis "Statistics". Right below there are icons that allow you to quickly do things that are very often done in Stata such as opening data, saving data, opening a log, browsing data, editing data, opening a do file etc. Stata is composed of four windows. The one on the upper left is a command review window where commands will be stored for the current session (but not saved once you have exited the current Stata session). The window on the lower left displays the variables in the

data that you are using. The upper right window displays results. The lower right window is the one you use to manually type commands.

Stata is a command-driven package. You can enter commands in either of three ways:

1. Manually: you type the first command in the command window and execute it, then the next, and so on.
2. Do-file: type up a list of commands in a “do-file”, essentially a computer programme, and execute the do-file.
3. Interactively: you click through the menu on top of the screen

The best way to learn Stata is by typing in the commands, so for this class you are expected to enter commands manually or use a do-file. Working interactively is useful when you are doing exploratory work. When you decide on your approach, you can then add the appropriate command to your do-file. The do-file is a permanent record of all your commands. **It is critical that you keep a record of your commands (in a do-file) to ensure you can re-produce your results.** Do-files are also useful if you want to make changes to your analysis (such as adding a regressor to your model) as you simply make the change in the do-file and re-run it. When the exact syntax of a command is hard to get right, use the menu-commands to do it once and then copy the syntax that appears in the results window into your do-file.

III. Getting Started

1. Saving your results – the LOG file

The results window displays your most recent commands and results but does not save. If you want to be retrieve the results from your *stata* session, you need to save them in a **log file**. I recommend opening a log-file at the start of each Stata session. Note that **you will need to change the drive and directory in which you are saving files to a location on your own computer**. Similarly, you will need to change the location from which you are reading files to the directory in which you have saved them (on your computer).

To open a log-file for today’s session, type the following in the command window:

```
log using "C:\MyDirectoryName\StataLab.log", replace
```

This command should be inserted at the start of your do-file. The command creates a log file named “StataLab” that records all your commands and results. The log file is saved onto your USB stick.

2. Reading in data

Let’s start with “WAGE.dta” The extension of a **Stata data set** is .dta and datasets in this format can be easily read in using the command `use`.

There are three different ways of reading this data into Stata: interactively using the drop down menus; typing in a command in the command window; executing a command form a do-file.

To open the data interactively, click on File (upper left), and click on Open → Computer → the directory in which you saved the data
Double click on the data file.

Once the data is loaded, the variables window will list all the variables in the data and the results window will show the following executed command:

```
use "C:\MyDirectoryName\WAGE.dta"
```

You could also open the data manually typing in the commands window the command:

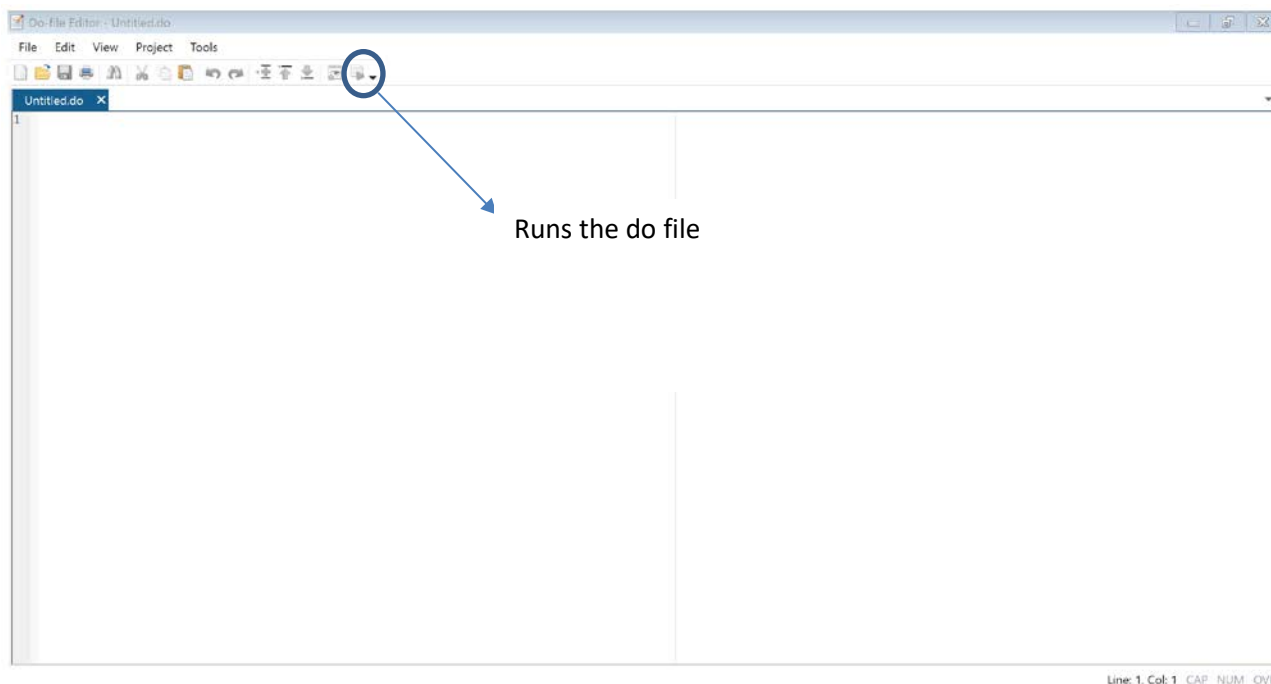
```
use "C:\MyDirectoryName\WAGE.dta", clear
```

The `clear` option will clear the dataset currently in memory before opening the other one.

We will now start a do file and read in the data by executing the do file.

3. Keeping a record of your commands: do-files

The do-file icon looks like a notebook with a pen, and it is located immediately below the “Statistics” menu. Double click on the icon to open a do-file and get the following window:



As soon as you open a do-file, write a note that describing what the program does and when it was created. You can write notes in do-files by typing the note like this: `/*[note]*/`. In the first lines of the do-file type:

```
/*Introduction to stata*/
```

```
/*This program runs a regression of lnwages and plots predictions*/
```

We will now open a log from the do-file. To close the log that we have already opened, type in the command window:

```
log close
```

Go back to the do-file and in the 4th line type:

```
log using "C:\MyDirectoryName\StataLab.log", replace
```

Logs can be appended or replaced. In the above command, I have used the option `replace`. Use the `append` option to add to the previous session. If you want to over-write the log each time you run the program, use the `replace` option.

Hint! In Stata the optional syntax of any command always follows a comma.

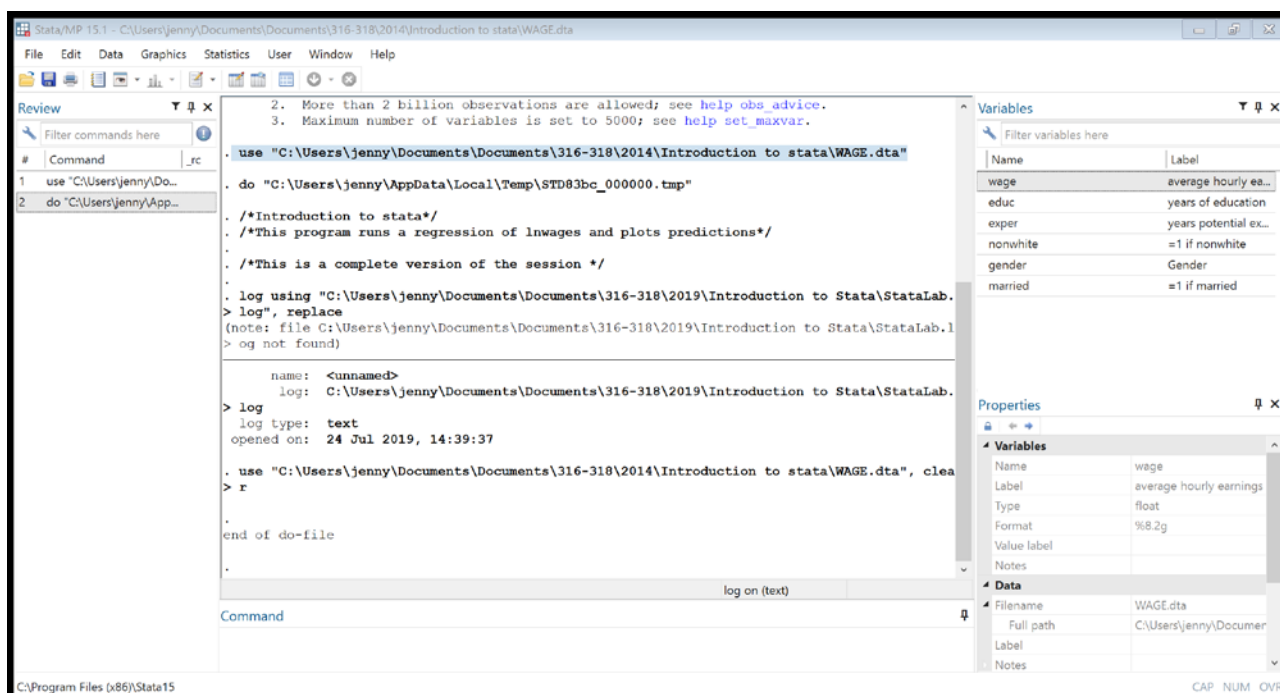
In the fifth line, type (or copy and paste from the results window):

```
use "C:\MyDirectoryName\WAGE.dta", clear
```

and hit enter.

1. Then click on the button that “runs the do-file”
2. Go back to stata’s main view.

The results window should display something like:



Now add a line to the do file and then enter the following command:

```
clear
```

and run that command. This clears the data in memory (check by going back to *stata*’s main view and look at the variables in memory window).

The variables window should be empty as the command “clear”, cleared the data from Stata’s memory.

Now we will read in an **excel data** from the do-file. To do that:

1. Go back to your do-file
2. In the next line of your do file type

```
import excel using "C:\MyDirectoryName\WAGE.xlsx"
```

What do you see in the variables in memory window? Browse the data using the data editor. What has happened?

Clear the data from memory and go back to your do file and append the last command as follows:

```
import excel using "C:\MyDirectoryName\WAGE.xlsx", firstrow
```

Check the variables in memory window. Better?

If the data that is in a **text file**, you can read this data into *stata* by first saving it as a txt file in excel. To read the txt file into *stata*:

1. Go back to your do-file
2. In the next line of your do file type

```
insheet using "C:\MyDirectoryName\WAGE.txt", clear
```

Select the line that you just typed in and hit the run button. Selecting is useful when you want to run only specific sections of a do-file. In this case, the insheet command.

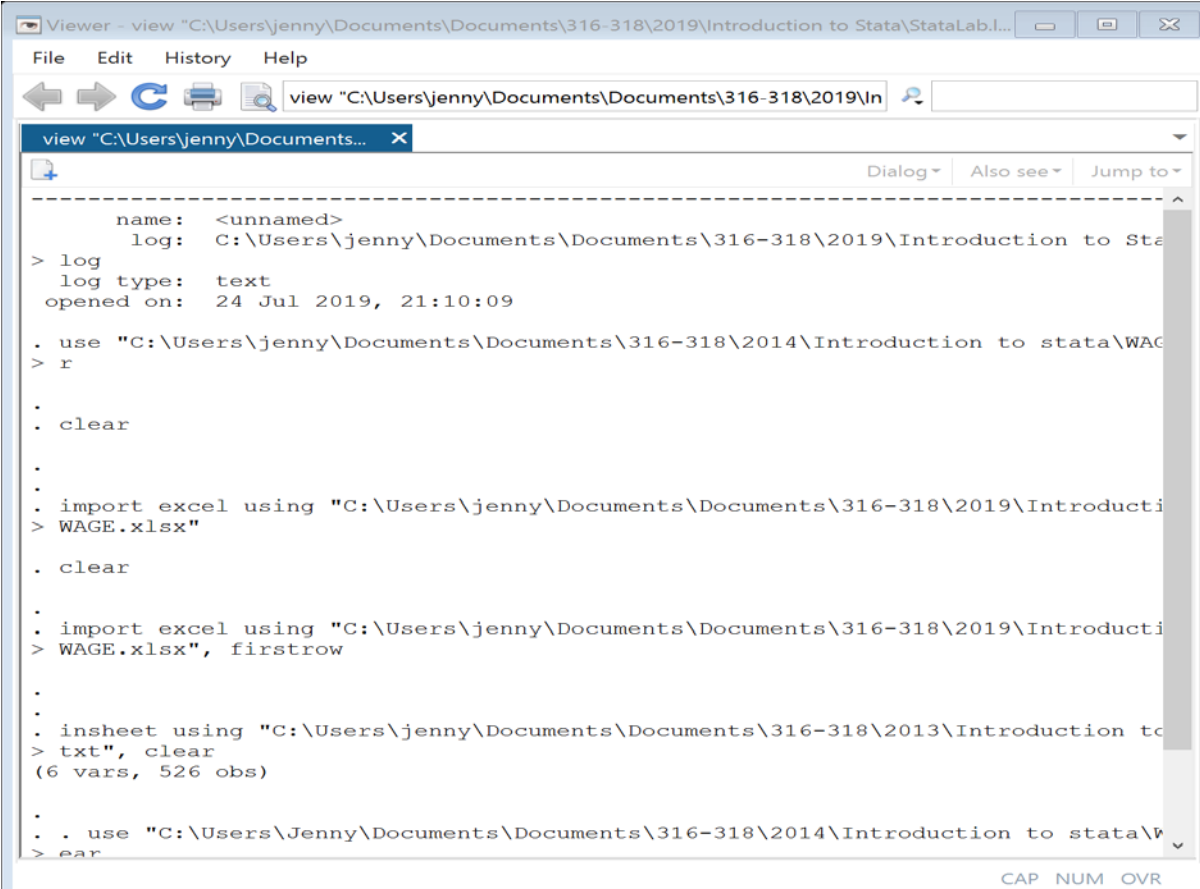
If you go back to *Stata*'s main view, the excel dataset should have been downloaded into *Stata*.

Let's now read in the *Stata* data (.dta data) from the do-file. To do so type the following in the do-file:

```
use "C:\MyDirectoryName\WAGE.dta", clear
```

Select this line in the do-file and hit the run button so that *Stata* reads in the .dta data.

Go back to *Stata*'s main window. If you click on File -> Log -> View, you will be able to see the entire list of commands that you have typed/ran since you opened the log file. The log file is a floating window that looks like this:



The screenshot shows a 'Viewer' window titled 'view "C:\Users\jenny\Documents\Documents\316-318\2019\Introduction to Stata\StataLab.I...'. The window contains a list of Stata commands and their output. The commands include: 'log', 'log type: text', 'opened on: 24 Jul 2019, 21:10:09', '. use "C:\Users\jenny\Documents\Documents\316-318\2014\Introduction to stata\WAGE.dta"', '. clear', '. import excel using "C:\Users\jenny\Documents\Documents\316-318\2019\Introduction to Stata\WAGE.xlsx"', '. clear', '. import excel using "C:\Users\jenny\Documents\Documents\316-318\2019\Introduction to Stata\WAGE.xlsx", firstrow', '. insheet using "C:\Users\jenny\Documents\Documents\316-318\2013\Introduction to Stata\WAGE.txt", clear', and '. use "C:\Users\Jenny\Documents\Documents\316-318\2014\Introduction to stata\WAGE.dta", clear'. The output for the last command shows '(6 vars, 526 obs)'. The window has a menu bar with 'File', 'Edit', 'History', and 'Help'. There are also navigation buttons and a search bar at the top.

```

name: <unnamed>
log: C:\Users\jenny\Documents\Documents\316-318\2019\Introduction to Stata
> log
log type: text
opened on: 24 Jul 2019, 21:10:09

. use "C:\Users\jenny\Documents\Documents\316-318\2014\Introduction to stata\WAGE.dta"
> r

.
. clear
.
.
. import excel using "C:\Users\jenny\Documents\Documents\316-318\2019\Introduction to Stata\WAGE.xlsx"
> WAGE.xlsx"

. clear

.
. import excel using "C:\Users\jenny\Documents\Documents\316-318\2019\Introduction to Stata\WAGE.xlsx", firstrow
> WAGE.xlsx", firstrow

.
. insheet using "C:\Users\jenny\Documents\Documents\316-318\2013\Introduction to Stata\WAGE.txt", clear
> txt", clear
(6 vars, 526 obs)

.
. use "C:\Users\Jenny\Documents\Documents\316-318\2014\Introduction to stata\WAGE.dta", clear
> ear
  
```

click on the X to close the file.

IV. Simple data management

Our next task is to estimate the following regression model with these data:

$$\ln W_i = \beta_0 + \beta_1 Educ_i + \beta_2 Experience_i + \beta_3 Experience_i^2 + \beta_4 Female_i + \beta_5 NonWhite_i + \beta_6 Married_i + \beta_7 Married_i * Female_i + u_i$$

$\ln W_i$ is the natural logarithm of wages of individual i . β_j $j=1,2,\dots,6$ are parameters to be estimated and u_i is the error term containing unobserved factors that affect wages. The idea of running this regression is to estimate the (statistical) effect of each covariate on the log of wages. For example, we may wish to know the effect of gender on wages, the effect of marriage on wages, and whether the effect of marriage is different for males and females.

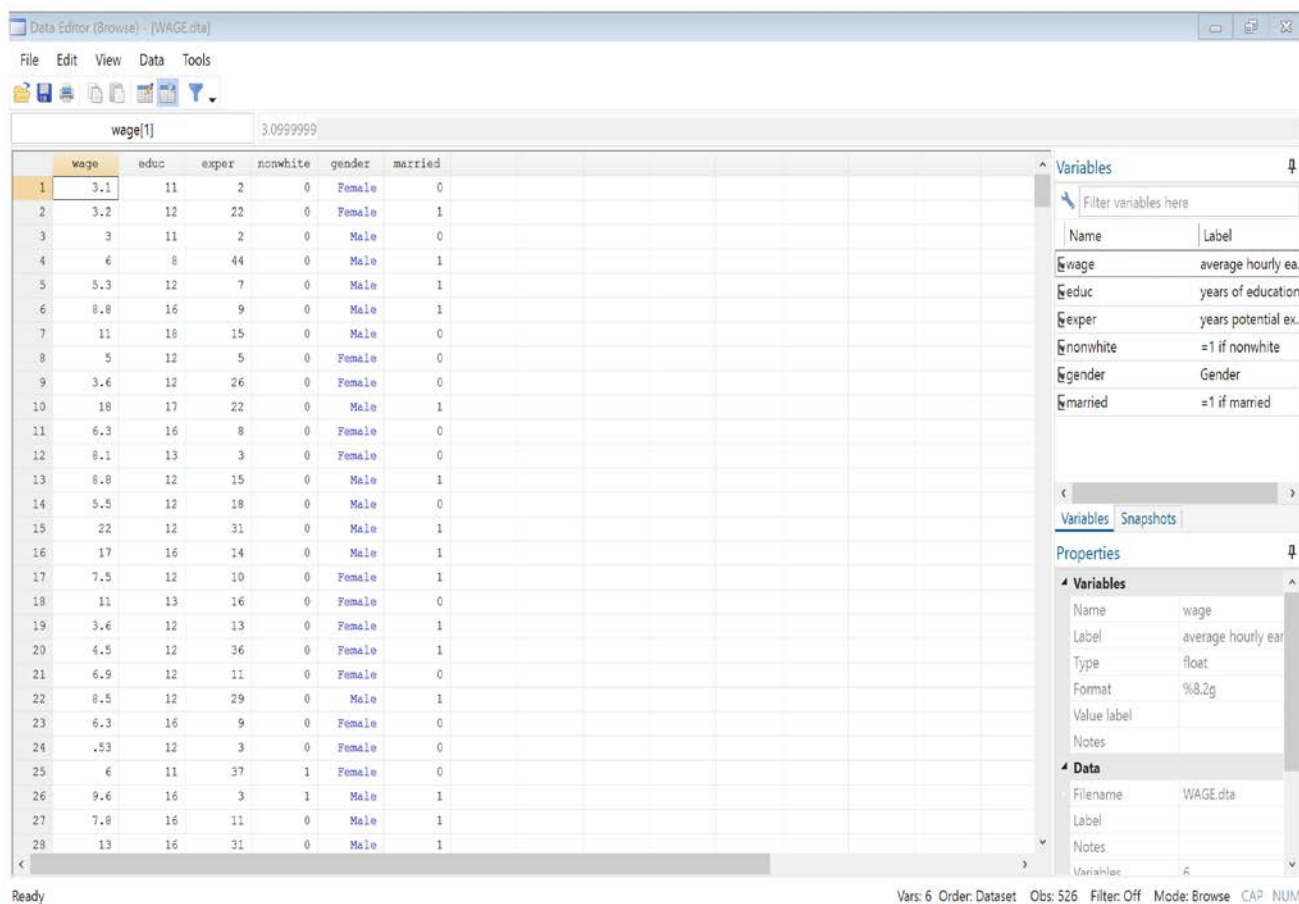
Before estimating a model, it is always important to gain some familiarity with the data and look at the descriptive statistics.

browse

We can look at the data using the Data Browser. You can do this typing the command **browse**

Alternatively, click on the data browser button .

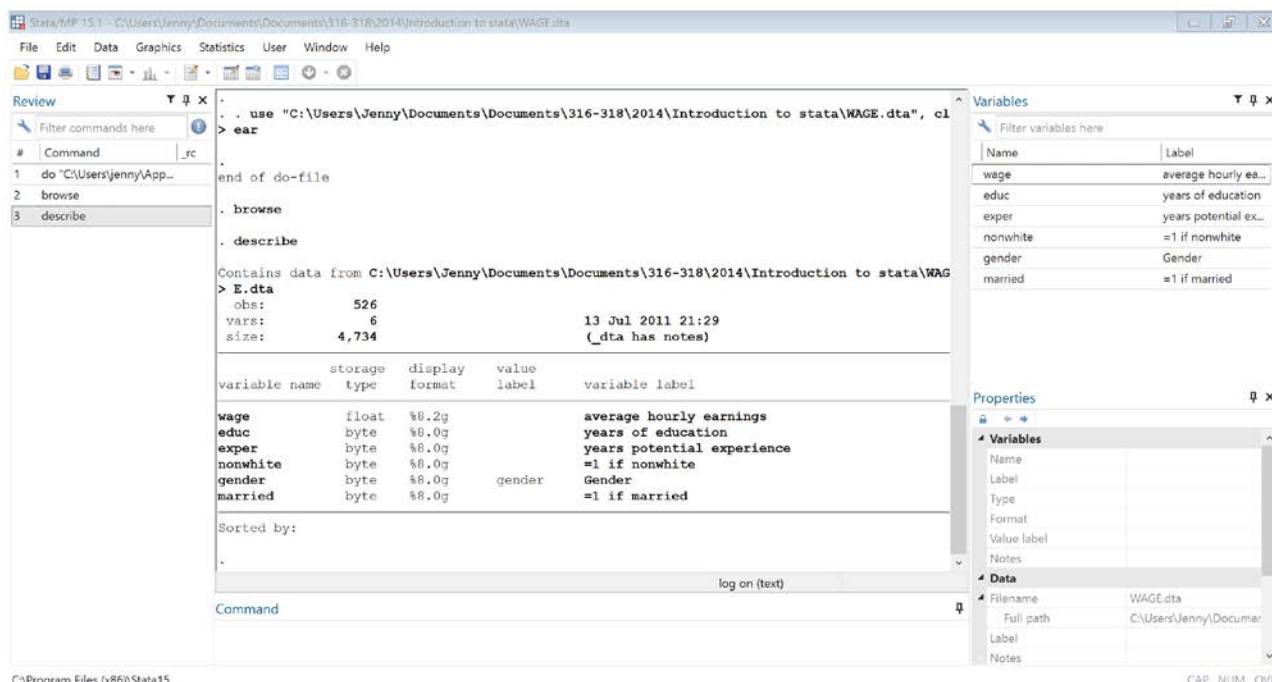
You will see the data as one rectangular table. The columns represent the variables and the rows represent the observations. The variables have descriptive names and the rows are numbered.



	wage	educ	exper	nonwhite	gender	married
1	3.1	11	2	0	Female	0
2	3.2	12	22	0	Female	1
3	3	11	2	0	Male	0
4	6	8	44	0	Male	1
5	5.3	12	7	0	Male	1
6	8.8	16	9	0	Male	1
7	11	18	15	0	Male	0
8	5	12	5	0	Female	0
9	3.6	12	26	0	Female	0
10	18	17	22	0	Male	1
11	6.3	16	8	0	Female	0
12	8.1	13	3	0	Female	0
13	8.8	12	15	0	Male	1
14	5.5	12	18	0	Male	0
15	22	12	31	0	Male	1
16	17	16	14	0	Male	1
17	7.5	12	10	0	Female	1
18	11	13	16	0	Female	0
19	3.6	12	13	0	Female	1
20	4.5	12	36	0	Female	1
21	6.9	12	11	0	Female	0
22	8.5	12	29	0	Male	1
23	6.3	16	9	0	Female	0
24	.53	12	3	0	Female	0
25	6	11	37	1	Female	0
26	9.6	16	3	1	Male	1
27	7.8	16	11	0	Male	1
28	13	16	31	0	Male	1

describe

The command `describe` allows us to see the structure of the dataset by describing its contents. Type `describe` in the commands window to get the following:



At the top of the listing, some information is given about the dataset such as its location, the memory it occupies, etc.

Then we have the variable names, storage type, format, value label and variable label.

For now we will focus on storage type. For our purposes it is enough to know that types beginning with `str` are string (text variables such as one's name) and all other variables are numeric. All the variables in our data are numeric.

We will get back to variable names and labels in a bit.

list (Syntax: `list [varlist] [if] [in] [, options]`)

The `list` command lists the values of the variables. For example typing:

```
list in 1/10
```

will display in the results window the values of all the variables of the first 10 observations in the data.

sort (Syntax: `sort varlist [in] [, stable]`)

There are many commands in Stata that require sorting a variable. The command **sort** arranges the observations of the current data into ascending order based on the values of the variables in `varlist`. For example, type `sort wage` to get in the results window:

```
.sort wage
```

Apparently Stata hasn't really done anything. However, if you type `browse`, you will see that the first observation in the data has the lowest value of wage, the second the next one and so on. You can click on X to close the data.

summarize

To obtain summary statistics for the data, you use the command “**summarize**” or “**sum**”. This will produce a table of summary statistics containing the Variable name, the number of observations, mean, standard deviation, minimum value and maximum value. If you want to get summary statistics for a specific variable or list of variables [varlist], type “`sum [varlist]`”

For example, type

```
sum wage
```

To summarize wages by gender, type in the following:

```
bysort gender: sum wage
```

you will see that the mean wages for females are lower than average wages for males.

An alternative way to subset the data on wages by gender is to use the “if” qualifier. The if qualifier uses a logical expression to determine which observations to use. If the expression is true, the observation is used in the command, otherwise it is skipped. Some operators whose results are either true or false are:

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

For example, to summarize wages for males type:

```
sum wage if gender==1
```

To summarize wages for females you can either type:

```
sum wage if gender==2
sum wage if gender!=1
```

Since gender only takes the values 1 and 2, the later command will produce descriptive statistics for wage when gender is different from 1 (males). That is, for females.

tabulate Syntax: **tabulate** varname [if] [in] [weight] [, tabulate1_options]

`tabulate` produces one-way tables (or two way cross-tabulations) of frequency counts. This is useful if you want to see the distribution of a variable (or the joint distribution of two variables). For example, to see the distribution of values in the schooling variable type

```
tabulate educ
```

or `tab educ`, for short.

Focusing your attention in the “Cum” column will tell you that 10.84 % of the sample have 9 or less years of education.

```
tab educ nonwhite
```

produces a cross-tabulation of educ and nonwhite. It shows the joint frequency distribution of these two variables.

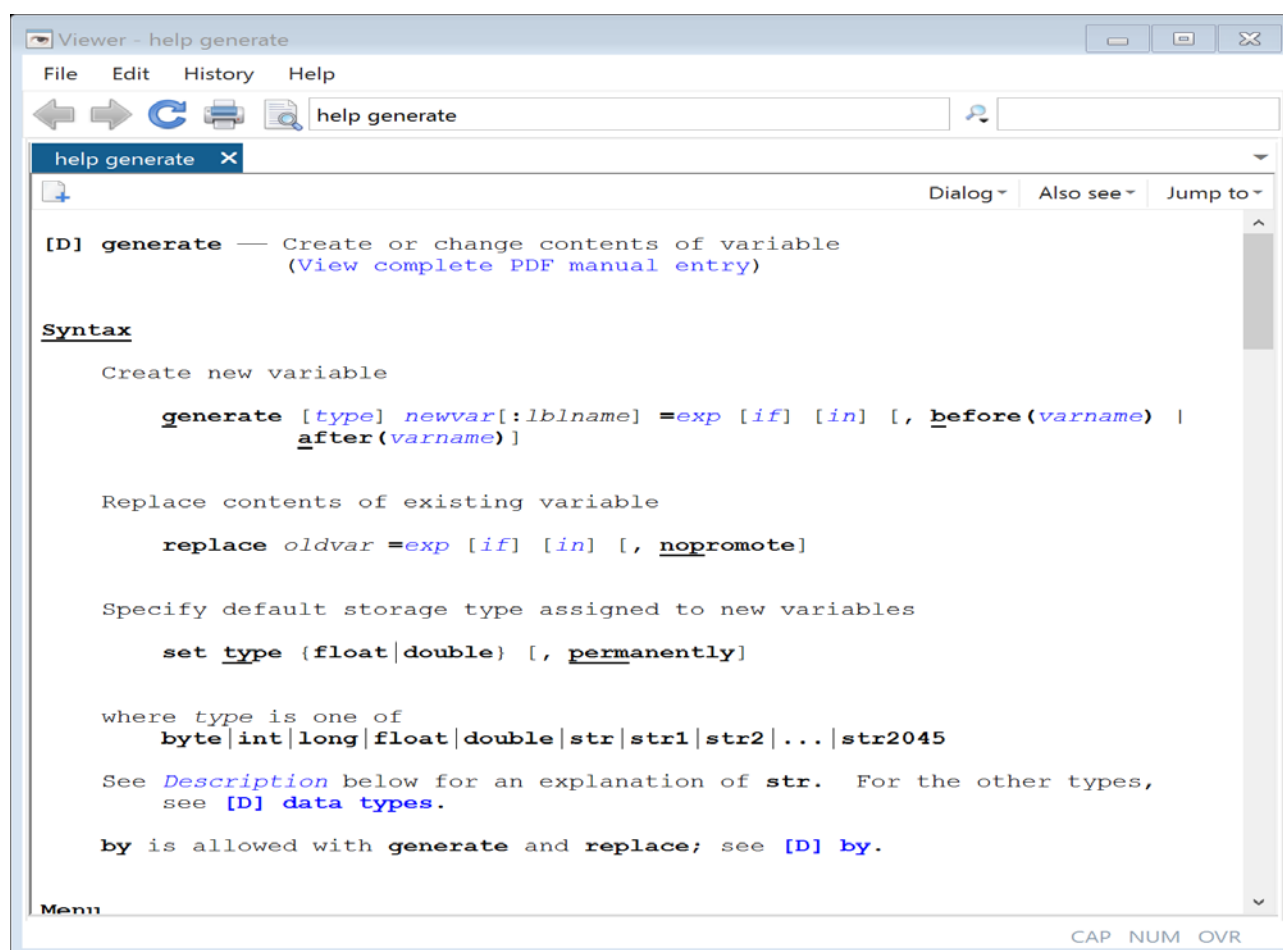
Hint! `tab1 varlist` produces a one-way tabulation for each variable specified in varlist. To see how this works try `tab1 educ nonwhite`

help Syntax: `help [command_or_topic_name] [, nonew name(viewername) marker(markername)]`

If you happen to know the Stata command that you want to use, but you don’t know its syntax type “`help [command]`”. The `help` command displays help information about the specified command or topic. For example, the next command that we will learn is `generate`. To learn about the syntax, description and options of this command type

help generate

to get the following:



Hint! Stata’s help can also be accessed from Menu: Help > Stata Command...

generate

generate creates a new variable. The values of the variable are specified by =exp. For example, to create the log of wages type

```
gen lnwage=ln(wage)
```

Highlight the command and run it. To see the mean of the newly created variable type in the command window

```
sum lnwage
```

To label your variable simply type in the do file:

```
label var lnwage "Ln hourly earnings"
```

Highlight the command and run it.

Now, let's generate and label experience squared by typing in the commands window the following:

```
gen expersq=exper^2
label var expersq "years of potential experience squared"
```

Highlight the command and run it.

Hint! egen is an extension of gen. To see the syntax and options of this command type `help egen`

replace Syntax: `replace oldvar =exp [if] [in] [, nopromote]`

replace changes the contents of an existing variable. Because replace alters data, the command cannot be abbreviated. To see how replace works let's create the interaction between female and married as follows.

First note that in the syntax of the replace command, there is an [if]. if at the end of a command means the command is to use only the data specified. if is allowed with most Stata commands.

Our next task is to create an interaction variable from the variables gender and married, where this variable will be one if the person is both female and married.

Let's first tabulate gender

```
tabulate gender
```

You will see that the variable has labels. We need to see the numbers assigned to the labels. To see that type:

```
tabulate gender, nolabel
```

the code for females is 2 and the code for males is 1. We also want to check whether gender has missing values. To do that type:

```
tabulate gender, mis
```

This variable does not have missing values. If it did, the number of observations missing will have shown up in the tabulate with a dot.

Let's create a dummy variable that is 1 if the person is a female and 0 if the person is a male using the variable gender. Return to the do file and type the following commands:

```
gen female=1 if gender==2
replace female=0 if gender==1
tab female gender, mis
```

and then run the command. The cross tab of female and gender indicates that, as desired, female is one when gender is female and is zero when gender is male.

Hint! There is yet another way to generate a dummy variable from a dichotomous or categorical variable. You can type:

```
. gen femaleclon=(gender==2)
. tab femaleclon gender
```

femaleclon will take the value of 1 when gender takes the value of 2 and 0 when gender is different from 2. In this case gender only has another value, which is one. Therefore, femaleclon takes the value of 0 when gender is one.

CAUTION! If gender had missing values, femaleclon will take the value of 0 when gender is missing, which is obviously wrong.

Labeling each variable right after you generate it will save you lots of trouble.

You might also want to add labels to the codes of the values of variables. You should do this in 3 steps. The first step labels the variable, the second step defines value labels and the third assigns value labels to variables. The syntax is:

1. Label variable Syntax: label variable varname ["label"]
2. Define value label Syntax: label define lblname # "label" [# "label" ...] [, add modify replace nofix]
3. Assign value label to variables Syntax label values varlist [lblname|.] [, nofix]

To do this using female as an example type (in the do-file):

```
label var female "Gender -female=1"
label define female 1"female" 0"male"
label values female female
```

Highlight the command and run it. To see the variable with labels type:

```
tab female
```

We now want to create an interaction of female and married. We know that female is one if the person is a female. Let's tab married:

```
tab married, mis
```

Married is also one if the person is married and zero otherwise. The variable married has no missing values. Now let's see how many people are both female and married with a cross tab

```
tab married female, mis.
```

132 observations are both female and married. To create an interaction between married and female simply type:

```
gen femmarried=married*female
tab femmarried, m
```

Highlight the command and run it. As desired, 132 observations are coded as 1 in femmarried because they are both female and married. To check your code further you can cross tab female and married by the interaction term as follows:

```
bysort femmarried: tab female married
```

and you will see that femmarried is one when both female and married take the value of one.

Let's label our femmarried label. Go to the do file and type:

```
label var femmarried "Interaction female*married"
label define femmarried 1"marr*female" 0"nonmarr/nonfemale or both"
label values femmarried femmarried
```

To see the variable with labels type:

```
tab femmarried
```

codebook

codebook reports the variable names, labels, and data to produce a codebook describing the dataset. To see what this command does type

```
codebook femmarried
```

```
. codebook femmarried
```

femmarried	Interaction female*married
type: numeric (float) label: femmarried range: [0,1] units: 1 unique values: 2 missing.: 0/526 tabulation: Freq. Numeric Label 394 0 nonmarr/nonfemale or both 132 1 marr*female	

You can see all the variables codebook by typing “codebook”.

Hint! To check other statistics handled by Stata click on Statistics in the main menu.

V. Regression analysis

As stated, the ultimate goal of this session is to estimate the following model:

$$\ln W_i = \beta_0 + \beta_1 Educ_i + \beta_2 Experience_i + \beta_3 Experience_i^2 + \beta_4 Female_i + \beta_5 NonWhite_i + \beta_6 Married_i + \beta_7 Married_i * Female_i + u_i$$

A specific goal of this section is to learn to create a do-file (program) while we are working interactively. Each time we type a command in the command window, copy and paste it to your do file.

Let's start. To double-check that everything is in order, let's check the sample statistics of the independent variables by typing:

```
sum educ exper expersq female married nonwhite femmarried
```

And the dependent variable:

```
sum lnwage
```

Caution! Don't forget to copy-paste the commands to your do-file

return list (r-class commands)

The Stata commands that analyse the data but do not estimate parameters are r-class commands. R-class commands save their results in r(). The contents of r() vary with the command and are listed by typing return list. For example, to see what Stata saved after we executed the command "sum lnwage", type

```
return list
```

```
. sum lnwage
```

Variable	obs	Mean	Std. Dev.	Min	Max
lnwage	526	1.623268	.5315382	-.6348783	3.218076

```
. return list
```

scalars:

```

r(N) = 526
r(sum_w) = 526
r(mean) = 1.623268444558514
r(var) = .2825328598278982
r(sd) = .5315382016637169
r(min) = -.6348783373832703
r(max) = 3.218075513839722
r(sum) = 853.8392018377781

```

The results in r disappear when a subsequent r-class command is executed. You can, however, save the values of say, the mean, using the generate command.

```
gen meanlnwage=r(mean)
```

to check the values of meanlnwage type:

```
sum meanlnwage
label var meanlnwage "Mean ln wage"
```

Caution! Don't forget to copy-paste the commands to your do-file

regress Syntax: regress depvar [indepvars] [if] [in] [weight] [, options]

regress fits a model of depvar on indepvars using linear regression. To fit our model type

```
regress lnwage educ exper expersq female married nonwhite femmarried
```

to get the following in the results window:

```
. regress lnwage educ exper expersq female married nonwhite femmarried
```

Source	SS	df	MS	Number of obs =	526
Model	63.2963349	7	9.04233355	F(7, 518) =	55.08
Residual	85.0334165	518	.164157175	Prob > F =	0.0000
				R-squared =	0.4267
				Adj R-squared =	0.4190
Total	148.329751	525	.28253286	Root MSE =	.40516

lnwage	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]
educ	.0819335	.0068934	11.89	0.000	.068391 .0954761
exper	.0350106	.0051505	6.80	0.000	.0248921 .045129
expersq	-.0006223	.0001109	-5.61	0.000	-.0008402 -.0004044
female	-.1196665	.0573735	-2.09	0.037	-.2323798 -.0069531
married	.2456996	.056712	4.33	0.000	.1342859 .3571133
nonwhite	-.0326389	.0587342	-0.56	0.579	-.1480253 .0827476
femmarried	-.3457532	.0736962	-4.69	0.000	-.4905333 -.2009731
_cons	.290758	.1035021	2.81	0.005	.0874226 .4940934

.

Hint! familiarize yourself with the regress command by typing `help regress`. Pay particular attention to *postestimation* and the section entitled *saved results*.

ereturn (e-class commands)

Estimation commands are e-class (estimation-class) commands, such as regress. The results of these commands are stored in e(). You can see the contents of the saved results after estimation by typing:

```
ereturn list
```

predict syntax: `predict [type] newvar [if] [in] [, single_options]`

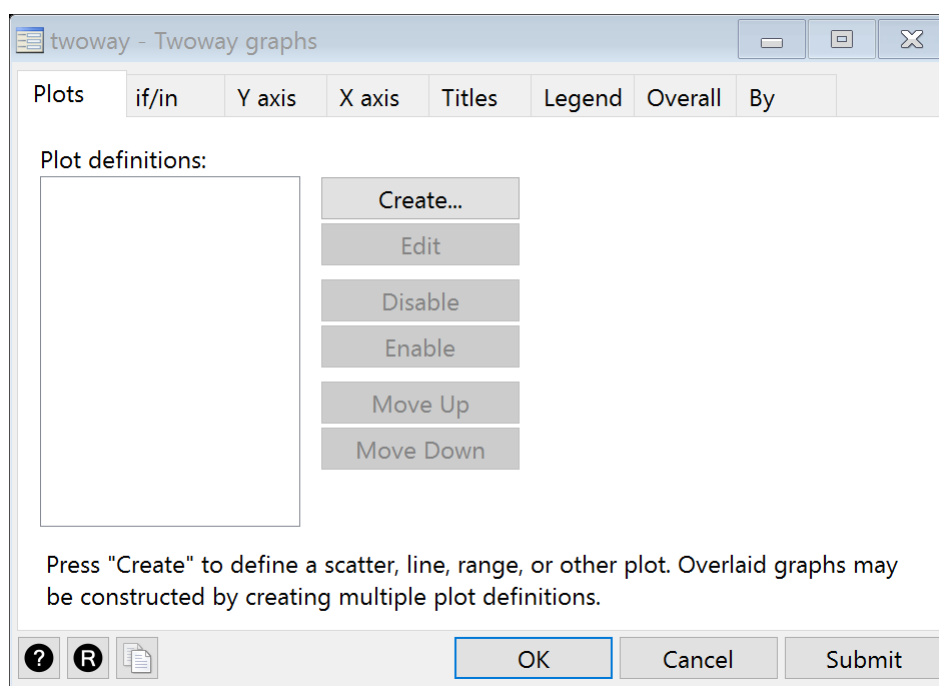
regress does not calculate residuals or predicted variables, but we can compute either of them after the regress command with predict. That is, predict obtains predictions, residuals, etc., after estimation. Predict computes the predicted values of the dependent variable by default. To get the predicted values of lnwage simply type `predict` followed by the name you are giving the predicted values of the dependent variable.

```
predict lnwagehat
label var lnwagehat "Predicted values ln wage"
```

Hint! predict has many options. For example to predict residuals you can type “predict residuals, resid”. This will save a variable named residuals that takes the values of the residuals.

graph

To evaluate the quality of the regression, let's graph the predicted values of lnwage vs the true values by typing. To do so, go to the main screen and click on graphics -> twoway graph to get:



Click on create and choose scatter. From the dropdown menu for the Y- variable, select `lnwagehat`. From the dropdown menu for the X-variable, select `lnwage`. Then click on Submit.

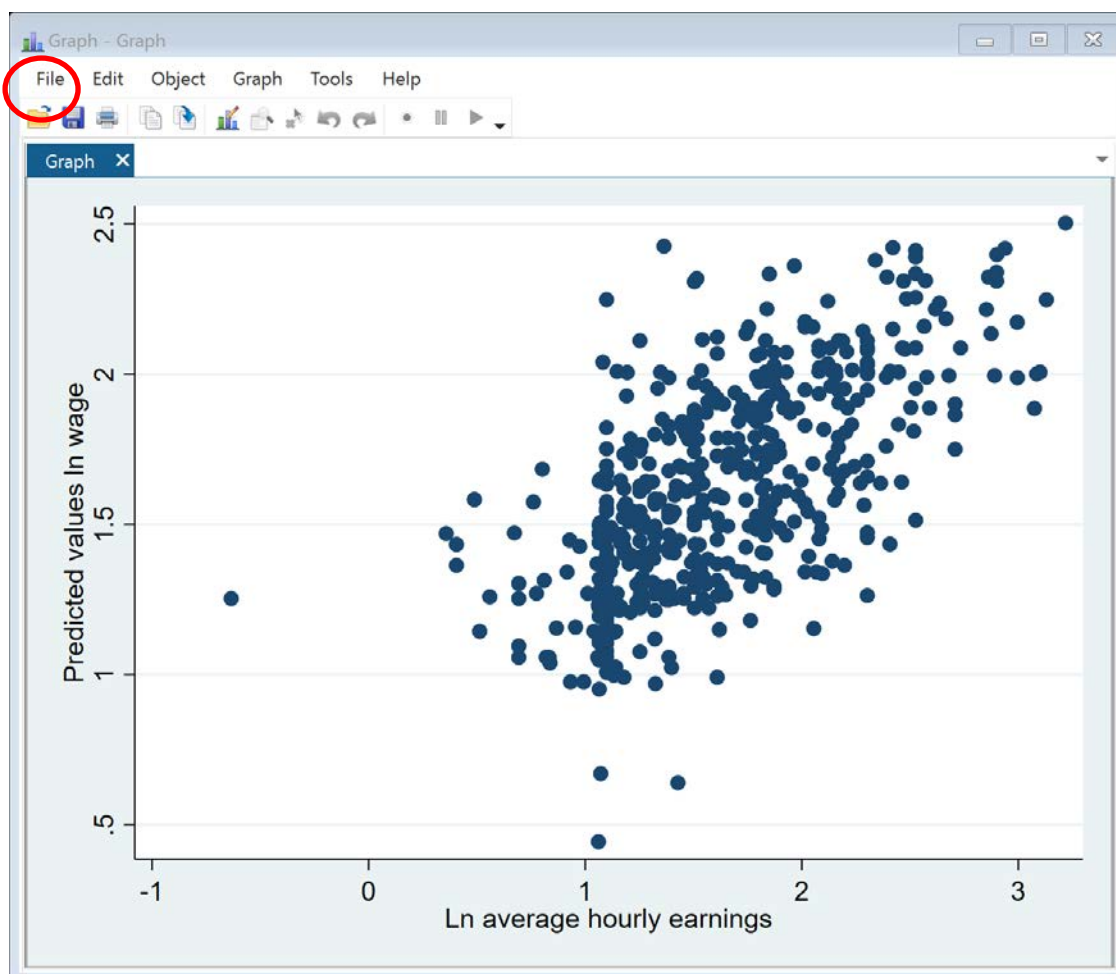
The graph will pop-up. Do you see any problems? What would you do to solve them?

Note that *Stata* has saved the command for the graph in the results window. Copy and paste the command in your do-file, so that you can replicate the graph later if you want:

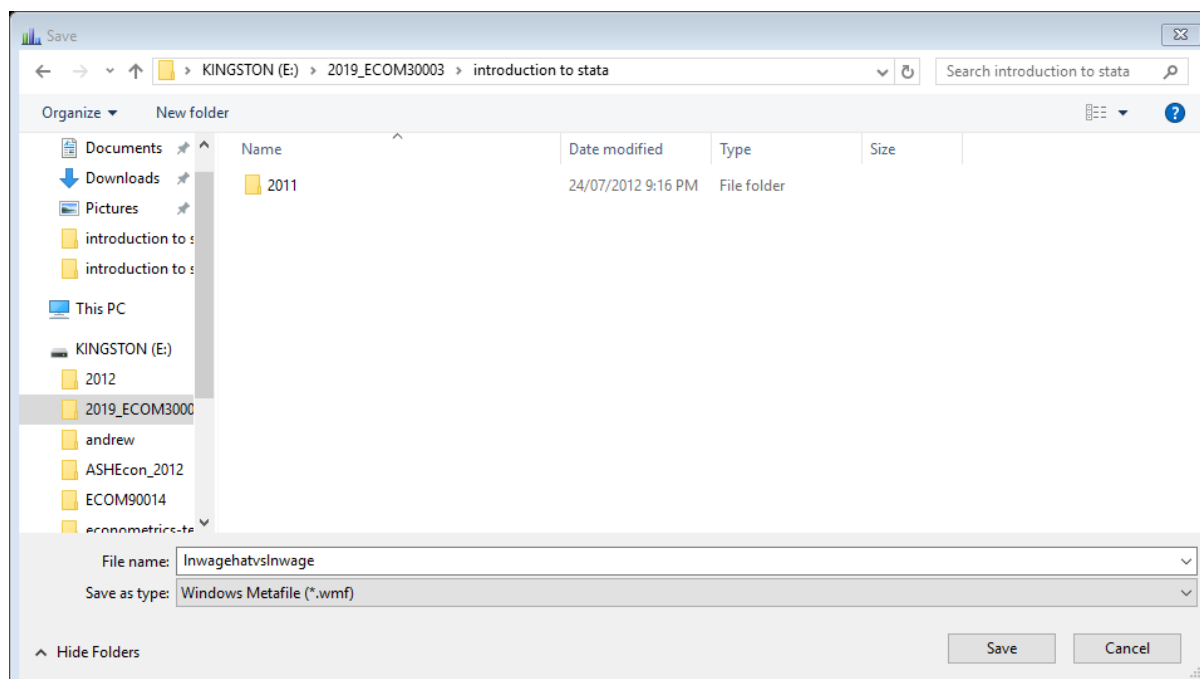
```
twoway (scatter lnwagehat lnwage)
```

Hint! type `help graph` to see Stata's graph options. `graph twoway` is widely used in applied econometrics. The command `graph save` will save a graph to your disk. The command `graph use` will load a graph that has been previously saved.

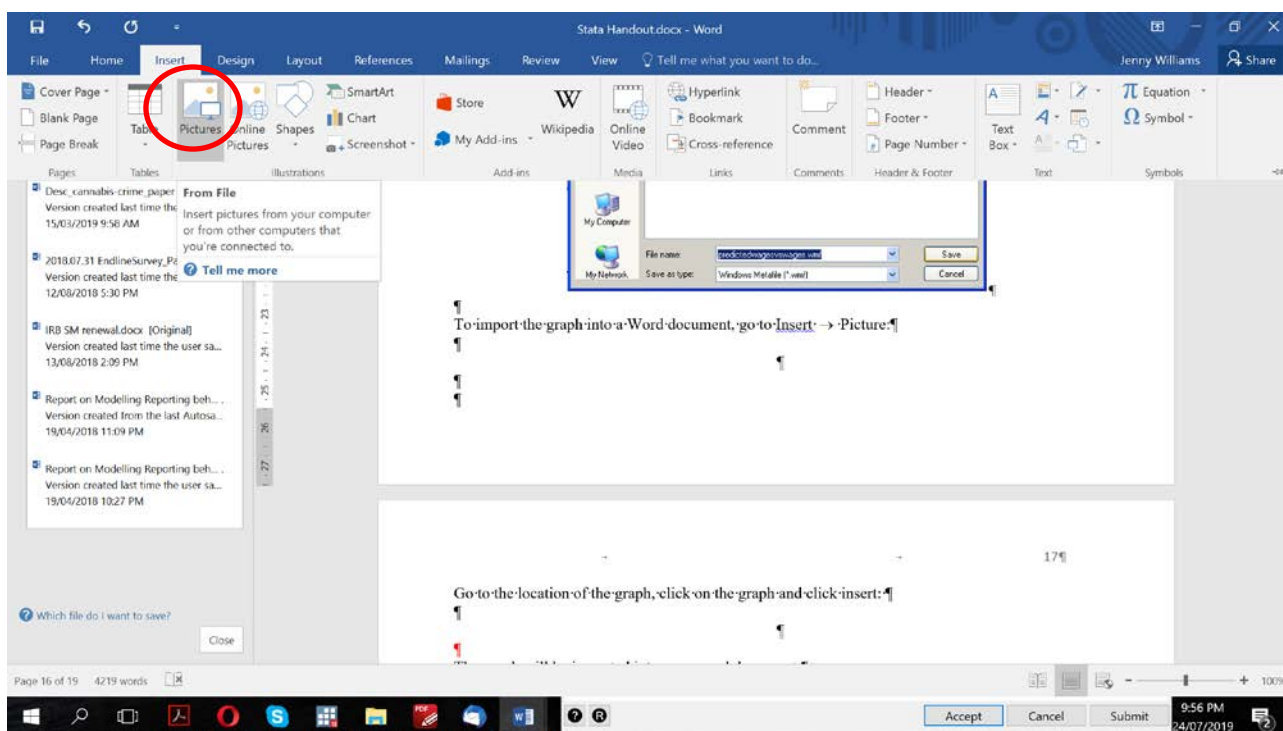
Hint! Stata graphs (saved as `.gph`) cannot be read into word. If you want to read graphs into word, from within the graph go to File and click on Save as:



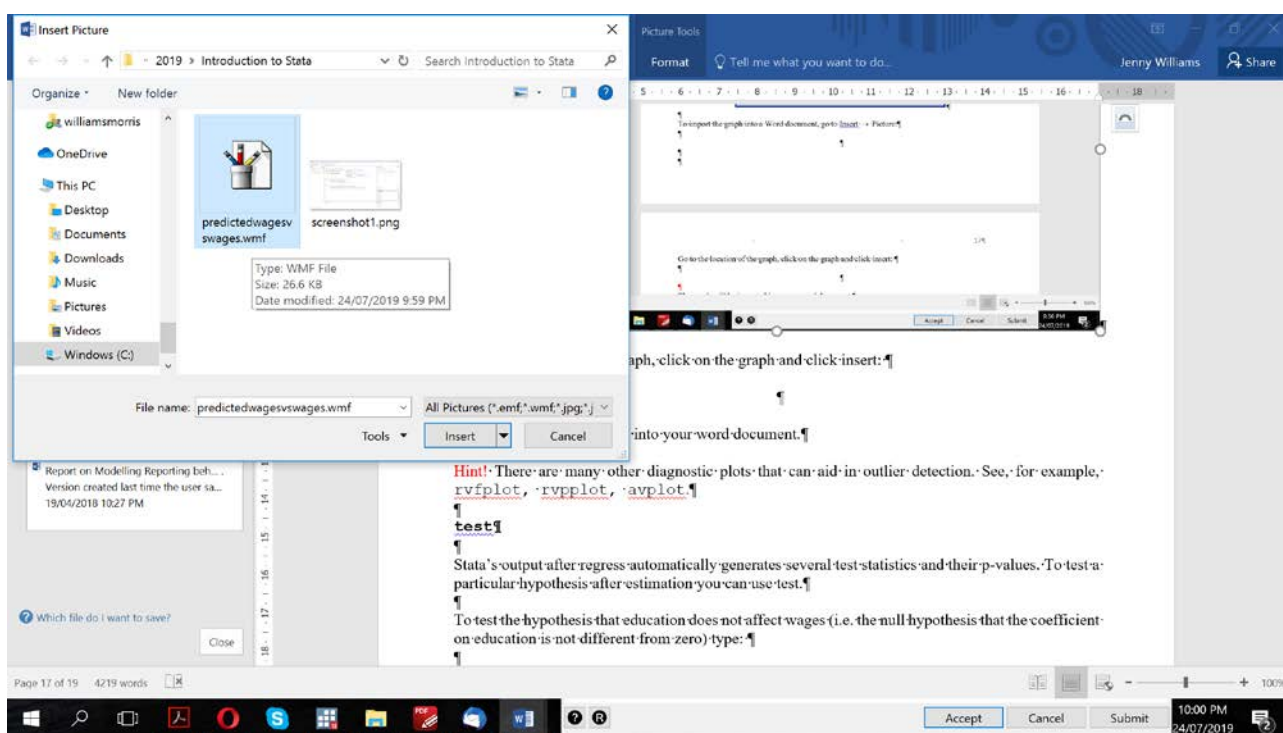
Save the graph as Windows Metafile (.wmf):



To import the graph into a Word document, go to Insert → Picture:



Go to the location of the graph, click on the graph and click insert:



The graph will be imported into your word document.

Hint! There are many other diagnostic plots that can aid in outlier detection. See, for example, `rvfplot`, `rvpplot`, `avplot`.

test

Stata's output after regress automatically generates several test statistics and their p-values. To test a particular hypothesis after estimation you can use test.

To test the hypothesis that education does not affect wages (i.e. the null hypothesis that the coefficient on education is not different from zero) type:

```
test educ
```

Based on an F of over 100 and a p-value of 0 .000 we can reject the hypothesis that the coefficient on education is equal to zero.

We'll now perform a joint test. We test the joint hypothesis that the coefficients on married and femmarried are jointly zero type:

```
test married femmarried
```

We next test the null hypothesis that being married does not affect the wages of females. That is, we test the hypothesis that the linear combination of female and the interaction of female and married is zero. To do so, type:

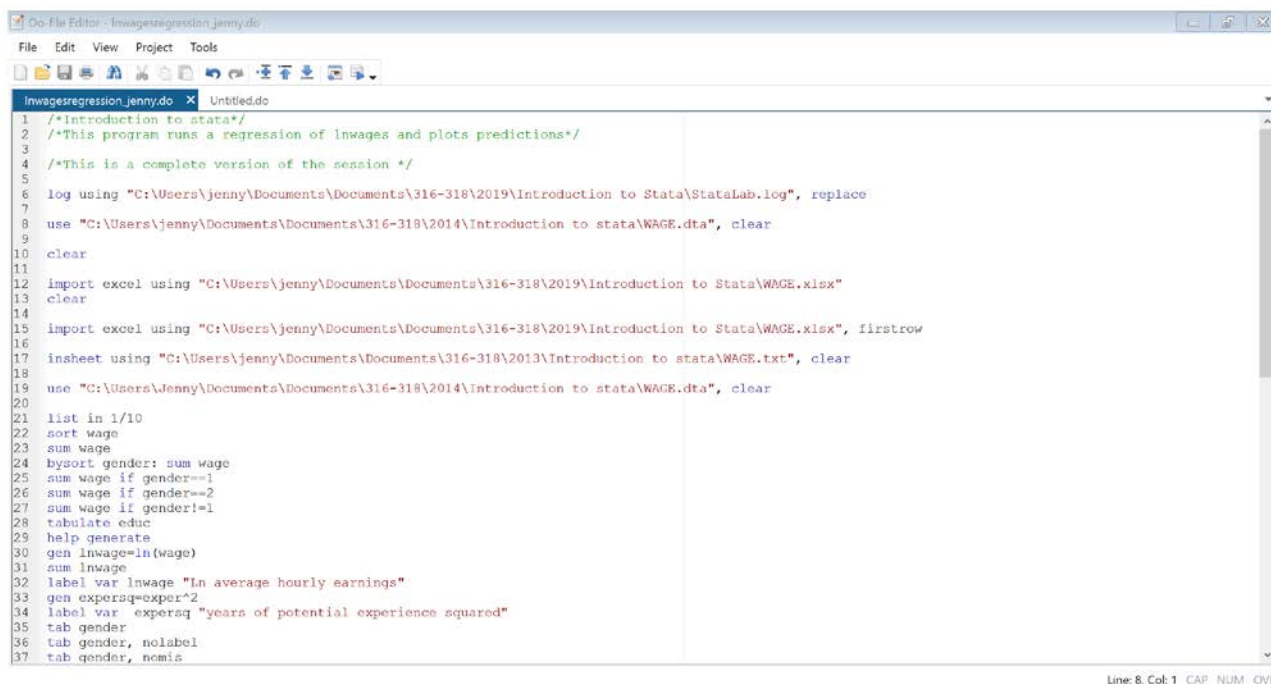
```
test married+femmarried=0
```

The F of 3.41 and the p-value of 0.0652 provide evidence that allows us to reject the hypothesis that the linear combination of female and the interaction term are zero.

Based on this evidence, what can you say about the association between being married and wages for females?

Back to the do file exercise

To save your program go to your USB stick. Your do-file program should look like this:



```

Do-File Editor - lnwagesregression_jenny.do
File Edit View Project Tools

1 /*Introduction to stata*/
2 /*This program runs a regression of lnwages and plots predictions*/
3
4 /*This is a complete version of the session */
5
6 log using "C:\Users\jenny\Documents\Documents\316-318\2019\Introduction to Stata\StataLab.log", replace
7
8 use "C:\Users\jenny\Documents\Documents\316-318\2014\Introduction to stata\WAGE.dta", clear
9
10 clear
11
12 import excel using "C:\Users\jenny\Documents\Documents\316-318\2019\Introduction to Stata\WAGE.xlsx"
13 clear
14
15 import excel using "C:\Users\jenny\Documents\Documents\316-318\2019\Introduction to Stata\WAGE.xlsx", firstrow
16
17 insheet using "C:\Users\jenny\Documents\Documents\316-318\2013\Introduction to stata\WAGE.txt", clear
18
19 use "C:\Users\Jenny\Documents\Documents\316-318\2014\Introduction to stata\WAGE.dta", clear
20
21 list in 1/10
22 sort wage
23 sum wage
24 bysort gender: sum wage
25 sum wage if gender==1
26 sum wage if gender==2
27 sum wage if gender!=1
28 tabulate educ
29 help generate
30 gen lnwage=ln(wage)
31 sum lnwage
32 label var lnwage "ln average hourly earnings"
33 gen expersq=exper^2
34 label var expersq "years of potential experience squared"
35 tab gender
36 tab gender, nlabel
37 tab gender, nomis
  
```

Line: 8, Col: 1 CAP NUM OVR

drop

In the do-file we are generating variables that were already generated interactively. These variables are `meanlnwage` `lnwagehat`. If we run the program now, we'll get an error. To solve this problem, we can use the command `drop`.

`drop` eliminates variables or observations from the data in memory. This command is not reversible so be sure to use it only when you know that you will not need the variables. Type:

```
drop meanlnwage lnwagehat
```

After you have run your program, to save the latest version of the data simply type:

```
save "C:\MyDirectoryName\WAGE_new.dta", replace
```

VI. Finishing a STATA session in the lab

At the end of each Stata session, make sure you do the following:

1. Type in the command window:

```
log close
clear
exit
```

The first line will end the recording of output. The second command will erase all content in Stata's memory space and the third command will close Stata.