

## Stata Tutoring Part I<sup>1</sup>

### Table of Contents

Memory Problems when Opening a File? .....	2
How to Set Your Working Directory .....	3
How to Read STATA Help.....	4
Opening an Existing Data Set: StatTransfer or Else.....	4
Creating a data set interactively .....	5
Descriptive statistics.....	6
Tables .....	8
Options “if”, “in” .....	10
Creating new variables .....	10
Dropping and Keeping Variables and/or Observations.....	13
Graphs.....	14
.do files .....	19
.log files.....	20
Some other useful commands .....	24
Regressions .....	26
xi expansions and interaction terms .....	29

---

<sup>1</sup> Prepared by Giovanni Oppenheim at Princeton University ([oppy@princeton.edu](mailto:oppy@princeton.edu)).

## Memory Problems when Opening a File?

In order to check the amount of memory you are allocating to data just do:

```
. memory
```

The output will be:

<b>Total memory</b>	<b>1,048,576 bytes</b>	<b>100.00%</b>
<b>overhead (pointers)</b>	<b>34,708</b>	<b>3.31%</b>
<b>data</b>	<b>260,310</b>	<b>24.83%</b>
-----		
<b>data + overhead</b>	<b>295,018</b>	<b>28.14%</b>
<b>programs, saved results, etc.</b>	<b>2,720</b>	<b>0.26%</b>
-----		
<b>Total</b>	<b>297,738</b>	<b>28.39%</b>
<b>Free</b>	<b>750,838</b>	<b>71.61%</b>

This tells you that you have 1,048 Kb allocated to memory. If your data set is large, and you need more memory, just do the following:

1. clear existing memory
2. increase the memory allocated to data

```
. clear  
. set memory 5m  
(5120k)
```

this will increase the memory allocated to data up to 5 Mb. In order to check, just do:

```
. memory
```

<b>Total memory</b>	<b>5,242,880 bytes</b>	<b>100.00%</b>
<b>overhead (pointers)</b>	<b>0</b>	<b>0.00%</b>
<b>data</b>	<b>0</b>	<b>0.00%</b>
-----		
<b>data + overhead</b>	<b>0</b>	<b>0.00%</b>
<b>programs, saved results, etc.</b>	<b>576</b>	<b>0.01%</b>
-----		
<b>Total</b>	<b>576</b>	<b>0.01%</b>
<b>Free</b>	<b>5,242,304</b>	<b>99.99%</b>

You could increase the memory allocated to data by a large amount, but always keep in mind that if you allocate too much memory to data the program will be much slower.

## How to Set Your Working Directory

Always check what your working directory is. You will always see your working directory in the bottom left corner of the Stata screen.

In order to explicitly check what your working directory is, the command and its output are:

```
. pwd  
C:\Stata
```

It is always safer to have a personal working directory where you store all the files you are using in the session. At the beginning of the session create a temporary folder in your hard disk, or on your floppy disk, and then store all the relevant files in it. For example, if you want to create a temporary folder on your desktop, set your cursor on the desktop, right click on it, and create a new folder (called “508”). This folder should appear as C:\WINDOWS\Desktop\508.

Most of the files we will be using will be posted in the shared folder “508b” on the server [WWS2k1](#). You will be able to access these files from any computer on campus, provided that you login with *your own* ID and PWD in the PRINCETON domain (it is very important that you login with your own ID, since access to these files is restricted to students taking the course; for example, if you’re working in the clusters and you are logged in as “Room 42 user”, you will not be able to access them. In this case, you’ll have to logout and login again with your own ID).

In order to access the files on the WWS 2k1 and copy them to your working directory, do the following:

1. In the Windows Start Menu click on “Run”.
2. In the command line type: [\\wws2k1\508b](#)

(or you can just follow the link from this file).

To copy the relevant files from the shared folder to your working directory, just drag them to your working directory. In our case I’ll ask you copy the following files:

```
CPS93.DTA  
Sat_scores.XLS.
```

In order to change your working directory from C:\Stata\ to C:\WINDOWS\Desktop\508\, execute, in Stata, the following command:

```
. cd c:\windows\desktop\508  
c:\windows\desktop\508
```

You can also achieve the same objective by simply opening Stata double clicking on the data set CPS93.DAT in the working directory.

Now your working directory is C:\WINDOWS\Desktop\508\.

You can check the content of the directory as in DOS:

```
. dir  
 <dir>  9/29/99 17:53  ..  
256.7k  9/24/98 17:49  CPS93.DTA  
15.0k   11/11/99 18:32  sat_scores.xls
```

The directory contains 2 files, a Stata data file (CPS93.dta) and an Excel data file (sat\_scores.xls).

## How to Read STATA Help

In order to ask for help, go to the *Help* menu in Stata and enter the command or the topic you're interested in. The green words are clickable and you can select the one you think is more appropriate. Otherwise you could type in the command line:

```
. help generate
```

```
-----  
help for generate, replace, seed                (manual:  [R] generate)  
-----
```

```
Create or change contents of variable  
-----
```

```
[by varlist:] generate [type] newvar = exp [if exp] [in range]  
[by varlist:] replace oldvar = exp [if exp] [in range] [,nopromote]  
set seed #  
set type { byte | int | long | float | double | str# }
```

```
Description  
-----
```

```
generate creates a new variable.  
replace changes the contents of an existing variable.  
--more--
```

This is the first screen of the “help” for command “generate”. The screen ends with  
--more--

you can scroll down by one line pushing “Enter”, or you could jump to the next screen by pushing the space bar or clicking the “Go” button in Stata. You cannot scroll back up. You can stop the scrolling of the screens by clicking the “Break” button (red dot). All items in the help line that are in square brackets are options to the command. All the others are necessary to perform the command. Usually “newvar” is the name of the new variable, “oldvar” is the name of the old variable, “exp” is logical expression (e.g. var1+var2), etc.

## Opening an Existing Data Set: StatTransfer or Else

If the existing data set is NOT in Stata format, use **StatTransfer**.

In our example, suppose you want to transfer the file sat\_scores.xls into Stata format, but you're interested only in some columns and some rows. When you open StatTransfer, you first have to select the **type** of the **input file** (in this case Excel), then you have to choose the file name (by browsing: in this case it is sat\_scores.xls). In order to select only some

variables (i.e. columns) check the menu *Variables* (also click on the *Optimize* option); in order to select only some observations (i.e. rows) check the menu *Observations* (remember to click on the option *preserve expression between transfers*). You can use logical expressions to select your observations. For example, you could decide to extract only observations related to males from a data set including both males and females: you would do this by specifying the expression *where sex = "male"* (if the original variable is *sex* and the value for male is *male*).

Once you've made the selection of the relevant variables and/or observations you want to transfer, you have to select the type of the **output file** (in this case Stata 6.0) and the name of the file. Then just click on *Transfer* and you're done. Now you have a data file in Stata format.

If the existing data set is in Stata Format, proceed as follows.

In the *File* menu in Stata select *Open...*, browse your system, and double click on the file you want to open. If you want to open C:\WINDOWS\Desktop\508\CPS93.dta, Stata will prompt:

```
. use "C:\WINDOWS\Desktop\508\cps93.dta", clear  
(March 93 CPS discouraged extrac)
```

The option **clear** tells you that Stata automatically erases all data in memory before opening a new file. If you have a data set in memory which was not saved, Stata prompts you with an alert signal:

*"Data in memory has changed since last save. Ok to clear current data. Cancel to abort"* and if you want to save your changes, just go "Cancel" and save.

To save, in the *File* menu select *Save* in order to save the data set with its existing file name, i.e. erasing the older file. Select *Save as...* in order to give a new name to the file you're working with.

## Creating a data set interactively

Click the "Editor" button and enter the Stata Editor. Then enter the values as you would do in Excel. In order to change the variable names from, say, "var1" to "age", double click the variable name in the gray area and change the name. Variable names cannot be longer than 8 characters.

When you're done, click *Preserve* and close the Stata Editor. Now you can save your data set.

## Descriptive statistics

To describe the data in memory do:

```
. describe
Contains data from C:\WINDOWS\Desktop\508\cps93.dta
  obs:           8,677           March 93 CPS discouraged extrac
  vars:           19             24 Sep 1998 17:49
  size:          277,664 (94.7% of memory free)
```

---

1. hhseq	long	%12.0g		Household sequence number
2. id	byte	%8.0g		ID type
3. lineno	byte	%8.0g		line number
4. age	byte	%8.0g		Age
5. marit	byte	%8.0g	maritlab	Marital Status
6. sex	byte	%8.0g	sexlab	Sex
7. educat	byte	%8.0g	higrad	Education
8. race	byte	%8.0g	racelab	Race
9. uslhrs	byte	%8.0g		usual weekly hours, mjob
10. earnhr	float	%9.0g		earnings per hour, mjob
11. wstat	byte	%8.0g	wstat	FTPT status
12. lfsr	byte	%8.0g	lfsrbl	labor force status recode
13. clswk	byte	%8.0g	clslab	class of worker
14. pid	byte	%8.0g		personal id in the family
15. fid	byte	%8.0g		family ID
16. fpersons	byte	%8.0g		# of persons in the famil
17. fownu6	byte	%8.0g		own children in family u6
18. state	byte	%8.0g	state	State
19. hhid	float	%12.0f		Household identif. number

---

Sorted by:

Stata will prompt you with the variable name (**sex**), the variable type (**byte**), the variable format (**%8.0g**), the value label (**sexlab**) and the variable label (**sex**). Variable labels are comments attached to variables. In order to attach to variable “age” the label “age at marriage” type:

```
. label var age "age at marriage"
. des age
  4. age      byte   %8.0g           age at marriage
```

(**des** stands for **describe**).

Value labels are labels attached to values of variables. Variable **sex** attains two values, 1 and 2. Label “male” is attached to value 1, and label “female” is attached to value 2. To see this type:

```
. label list sexlab
sexlab:
```

```
  1 male
  2 female
```

To create a value label (e.g. “sexlab”) and to attach it to values of a particular variable:

```
. label define sexlab 1 "male" 2 "female" (creates the label)
. label values sex sexlab (attaches it to variable “sex”)
```

Another way of inspecting data is to summarize them:

```
. summarize
```

Variable	Obs	Mean	Std. Dev.	Min	Max
hhseq	8677	51098.76	18534.47	27999	73871
id	8677	41.76985	.9587741	41	52
lineno	8677	1.812954	1.092296	1	36
age	8677	36.07065	11.67076	18	64
marit	8677	3.348047	2.730244	1	7
sex	8677	1.516768	.4997475	1	2
educat	8677	39.32223	2.237435	31	46
race	8677	1.188429	.4925585	1	3
uslhrs	8677	35.98859	10.15854	1	99
earnhr	8677	9.321139	5.137567	.5	69.99
wstat	8677	2.523222	.9558349	2	5
lfsr	8677	1.03976	.1954068	1	2
clswk	8677	1.322807	.8534107	1	4
pid	8677	1.765933	.9545703	1	10
fid	8677	1.072836	.2900628	1	6
fpersons	8677	3.071108	1.517727	1	14
fownu6	8677	.2605739	.5751343	0	5
state	8677	51.8299	26.43896	11	95
hhid	8677	4.34e+11	3.28e+11	5.13e+07	1.00e+12

In the first column you have the number of observations for which each variable is “not missing”. Then you have sample mean, sample standard deviation, minimum value in the sample and maximum value in the sample. You can also get percentile for any variable (e.g. age) by typing:

```
. summarize age, detail
```

```
age at marriage
```

Percentiles		Smallest		
1%	18	18		
5%	20	18		
10%	21	18	Obs	8677
25%	26	18	Sum of Wgt.	8677
50%	35		Mean	36.07065
		Largest	Std. Dev.	11.67076
75%	44	64		
90%	53	64	Variance	136.2067
95%	58	64	Skewness	.4004839
99%	62	64	Kurtosis	2.274776

## Tables

### One-way tables

```
. tab race
  Race |      Freq.      Percent      Cum.
-----+-----
  white |      7431      85.64      85.64
  black |       857       9.88      95.52
  other |       389       4.48     100.00
-----+-----
  Total |      8677     100.00
```

This gives you the absolute frequency for each value of the variable, the percent, and the cumulative distribution.

You could get the same table without the value label:

```
. tab race, nolabel
  Race |      Freq.      Percent      Cum.
-----+-----
     1 |      7431      85.64      85.64
     2 |       857       9.88      95.52
     3 |       389       4.48     100.00
-----+-----
  Total |      8677     100.00
```

### Two-way tables

```
. tab race sex
  Race | Sex      male      female |      Total
-----+-----+-----
  white |      3608      3823 |      7431
  black |       399       458 |       857
  other |       186       203 |       389
-----+-----+-----
  Total |      4193      4484 |      8677
```

In order to get row percentages type:

```
. tab race sex, row
  Race | Sex      male      female |      Total
-----+-----+-----
  white |      3608      3823 |      7431
        |      48.55      51.45 |     100.00
-----+-----+-----
  black |       399       458 |       857
        |      46.56      53.44 |     100.00
-----+-----+-----
  other |       186       203 |       389
        |      47.81      52.19 |     100.00
-----+-----+-----
  Total |      4193      4484 |      8677
        |      48.32      51.68 |     100.00
```

In order to get row percentages without absolute frequencies type:

```
. tab race sex, row nofreq
```

Race	Sex		Total
	male	female	
white	48.55	51.45	100.00
black	46.56	53.44	100.00
other	47.81	52.19	100.00
Total	48.32	51.68	100.00

In order to get column percentages and row percentages without absolute frequencies type:

```
. tab race sex, column row nofreq
```

Race	Sex		Total
	male	female	
white	48.55	51.45	100.00
	86.05	85.26	85.64
black	46.56	53.44	100.00
	9.52	10.21	9.88
other	47.81	52.19	100.00
	4.44	4.53	4.48
Total	48.32	51.68	100.00
	100.00	100.00	100.00

Sorted tables: In Stata you can sort (or "order") your data according to the values of one or more variables (as in Excel: see *Data/Sort...*). When you sort your data, the order of the variables you're sorting by is important. In order to sort using the variable **sex**, you just have to type:

```
. sort sex
```

Once you have sorted your data set by some variables, you can use the option **by** in order to get information for subgroups defined by all the possible combinations of the variables you sorted by. For example, in order to tabulate race by sex, first sort by sex and then tabulate race:

```
. sort sex
. by sex: tab race
```

-> sex= male

Race	Freq.	Percent	Cum.
white	3608	86.05	86.05
black	399	9.52	95.56
other	186	4.44	100.00
Total	4193	100.00	

-> sex= female

Race	Freq.	Percent	Cum.
------	-------	---------	------

white	3823	85.26	85.26
black	458	10.21	95.47
other	203	4.53	100.00
<b>Total</b>	<b>4484</b>	<b>100.00</b>	

## Options “if”, “in”

You can restrict your commands to certain specific observations.

Examples:

To get the race distribution of people aged 18-40:

```
. tab race if age>=18 & age<=40
```

Race	Freq.	Percent	Cum.
white	4912	85.74	85.74
black	551	9.62	95.36
other	266	4.64	100.00
<b>Total</b>	<b>5729</b>	<b>100.00</b>	

To get the sex distribution of white people (race=1), and of black people (race=2) aged 40+, type:

```
. tab sex if race==1 | (race==2 & age>=40)
```

Sex	Freq.	Percent	Cum.
male	3760	48.41	48.41
female	4007	51.59	100.00
<b>Total</b>	<b>7767</b>	<b>100.00</b>	

(“|” stands for “or” in Stata)

To get the race distribution of observations 345 to 543 type

```
. tab race in 345/543
```

Race	Freq.	Percent	Cum.
white	170	85.43	85.43
black	19	9.55	94.97
other	10	5.03	100.00
<b>Total</b>	<b>199</b>	<b>100.00</b>	

## Creating new variables

To create a new variable, e.g. log(earnhr), type

```
. generate lnw=log(earnhr)
```

To create a new variable that groups an existing “continuous” variable in a “grouped” variable (e.g. “age” in groups 18-25, 26-35, 36-45, 46-55, 56+) type:

```
. gen agegr=recode (age, 25, 35, 45, 55, 64)
```

(gen stands for generate)

```
. label var agegr "age grouped" (this command tells Stata to attach a name label to the newly created variable, and the label is age grouped)
```

In order to check your new variable you can do:

```
. tab age agegr
```

Age	age grouped					Total
	25	35	45	55	64	
18	202	0	0	0	0	202
19	212	0	0	0	0	212
20	252	0	0	0	0	252
21	257	0	0	0	0	257
22	263	0	0	0	0	263
23	282	0	0	0	0	282
24	261	0	0	0	0	261
25	226	0	0	0	0	226
26	0	231	0	0	0	231
27	0	245	0	0	0	245
28	0	273	0	0	0	273
29	0	247	0	0	0	247
30	0	273	0	0	0	273
31	0	219	0	0	0	219
32	0	289	0	0	0	289
33	0	237	0	0	0	237
34	0	264	0	0	0	264
35	0	268	0	0	0	268
36	0	0	251	0	0	251
37	0	0	240	0	0	240
38	0	0	264	0	0	264
39	0	0	205	0	0	205

--more--

As you can see, “agegr” assumes value 25 for all those aged 18-25, value 35 for all those aged 32-35, etc.

To create several “dummy” variables from a categorical variable (e.g. from “race”) type:

```
. tab race, gen(drace)
```

Race	Freq.	Percent	Cum.
white	7431	85.64	85.64
black	857	9.88	95.52
other	389	4.48	100.00
Total	8677	100.00	

This generates three new variables, “drace1”, “drace2”, and “drace3” with the following properties:

“drace1”=1 if “race”=1, “drace1”=0 if “race”≠1

“drace2”=1 if “race”=2, “drace2”=0 if “race”≠2

“drace3”=1 if “race”=3, “drace3”=0 if “race”≠3

In order to summarize these new variables, type:

```
. sum drace*
Variable |      Obs      Mean   Std. Dev.   Min   Max
-----+-----
drace1 |    8677    .856402   .3507019     0     1
drace2 |    8677    .0987669   .2983659     0     1
drace3 |    8677    .0448312   .2069451     0     1
```

And if you want to describe them, do:

```
. des drace*
23. drace1      byte   %8.0g                race==white
24. drace2      byte   %8.0g                race==black
25. drace3      byte   %8.0g                race==other
```

(the \* symbol indicates that you want to summarize, or to describe, all the variables with name beginning with **drace**). You can see that 85.6% of the sample is white, 9.9% is black and 4.5% is “other”.

For an alternative way of creating dummy variables from categorical variables, see the section on xi expansion on page 29.

You can also change the name of a variable in Stata by typing:

```
. rename uslhrs wage
```

here we renamed variable “uslhrs” and gave it the new name “wage”. Notice that the variable label is unchanged:

```
. des wage
9. wage          byte   %8.0g                usual weekly hours, mjob
```

You can also re-code some values for a specific variable. Suppose you want to change the coding for variable “sex”, and have it as follows:

“sex”=0 if “female”

“sex”=1 if “male”

Now your coding is:

“sex”=2 if “female”

“sex”=1 if “male”

All you have to do is type:

```
. recode sex 2=0
(4484 changes made)
```

In order to check, type:

```
. tab sex, nolabel
      Sex |      Freq.   Percent   Cum.
-----+-----
      0 |    4484    51.68    51.68
      1 |    4193    48.32   100.00
-----+-----
    Total |    8677   100.00
```

And compare the result with the previous one.

(Notice the option **nolabel**: since variable “sex” is labeled with label “sexlab” for values 1 and 2, if you don’t change the label your new value 0 will not be labeled after your replacement).

## Dropping and Keeping Variables and/or Observations

You can keep only some observations, and drop others. In order to show this, suppose you're interested only in the subsample of males. You could either drop all the females:

```
. drop if sex==2
(4484 observations deleted)
(we went back to the old coding of "sex")
```

or you could keep all the males:

```
. keep if sex==1
(4484 observations deleted)
```

As you can see the result is the same: 4,484 observations, i.e. all the females, were dropped from the sample.

You can also keep some variables, or drop some variables. Suppose you want to drop all the variables but age, sex, race. The fastest way to do is:

```
. keep age sex race
. des
Contains data from C:\WINDOWS\Desktop\508\cps93.dta
  obs:          4,193                March 93 CPS
discouraged extrac
  vars:          3                    24 Sep 1998 17:49
  size:          29,351 (97.0% of memory free)
```

---

1. age	byte	%8.0g		Age
2. sex	byte	%8.0g	sexlab	Sex
3. race	byte	%8.0g	racelab	Race

---

Sorted by:

Note: data has changed since last save

All other variables have been dropped from the data set in memory. Stata tells you that your data set has changed since last save.

**IMPORTANT:** If you want to be able to recover the dropped variables afterwards, you must be careful not to save this new data set (with only 3 variables) with the same name (i.e. in the same file) of the original one.

You could also want to drop only a few variables from the original data set. Suppose you're not interested in "number of children in the household": then

```
. des fown*
17. fownu6      byte    %8.0g                own children
in fam. u-6
18. fownu18     byte    %8.0g                onw never
marr. chil u-18
```

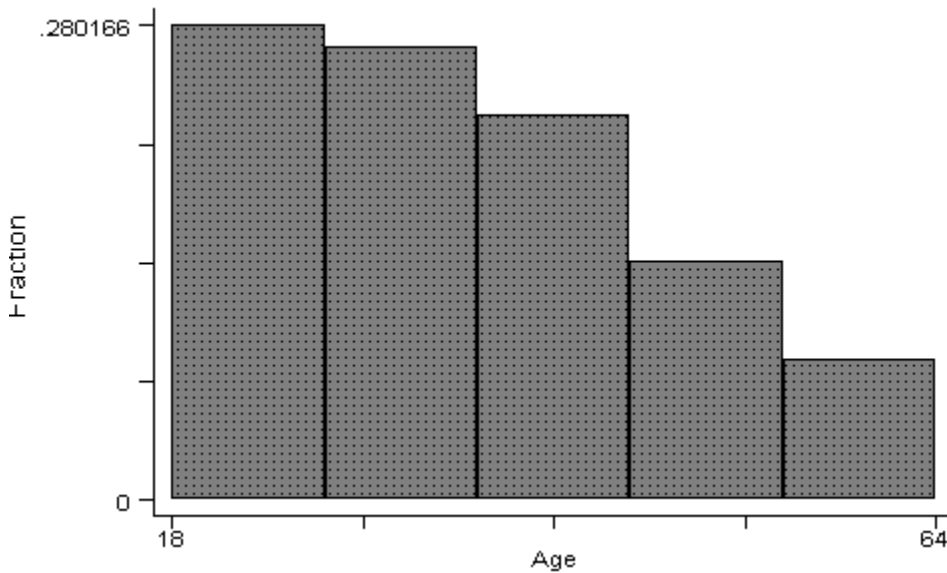
```
. drop fown*
```

(you dropped "fownu6" and "fownu18", the only two variable with name beginning by "fown").

## Graphs

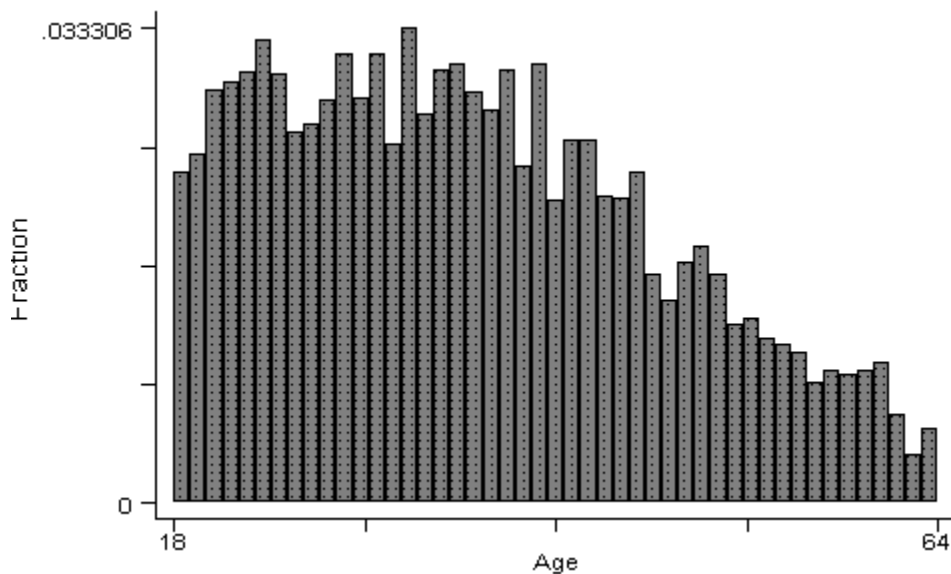
If you want to plot the histogram of a discrete variable, simply type:

```
. graph age
```



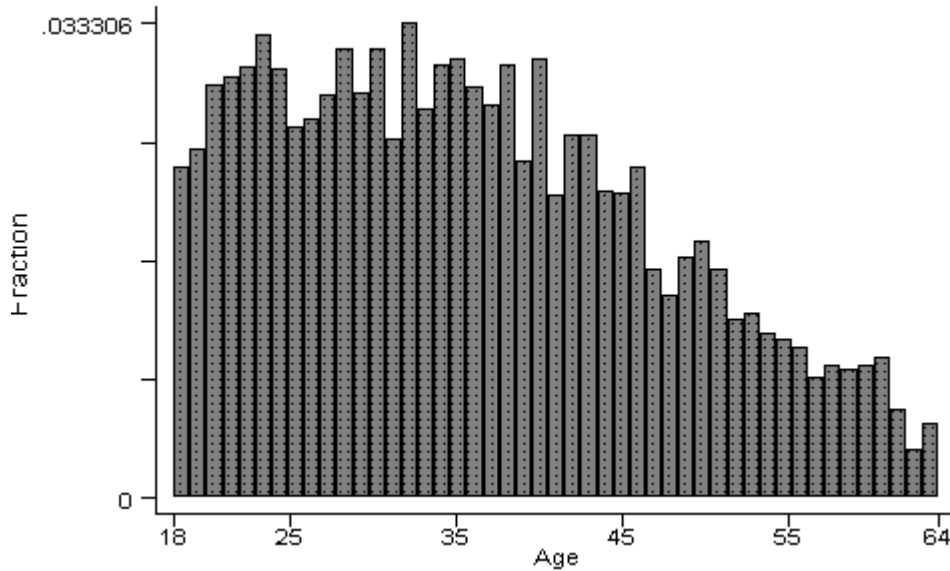
Unfortunately Stata automatically divides the support of the “age” distribution into 5 intervals. Since we have a discrete variable that can attain 47 values (i.e.  $64-18+1$ ), we might want to divide the support into 47 intervals. To do this, add the option `bin` :

```
. graph age, bin(47)
```



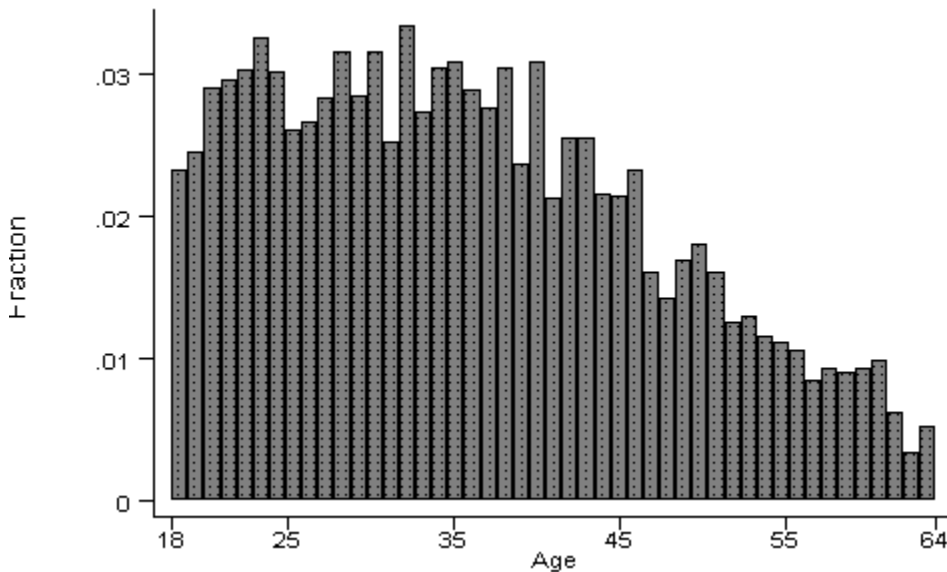
You might want to rescale the horizontal axis: Stata automatically enters only the minimum and the maximum of the support. You might want to enter also the following numbers: 25,35,45,55. Use the option `xlabel` (...):

```
. graph age, bin(47) xlabel(18,25,35,45,55,64)
```



You can also rescale the vertical axis, using the option `ylabel(...)`:

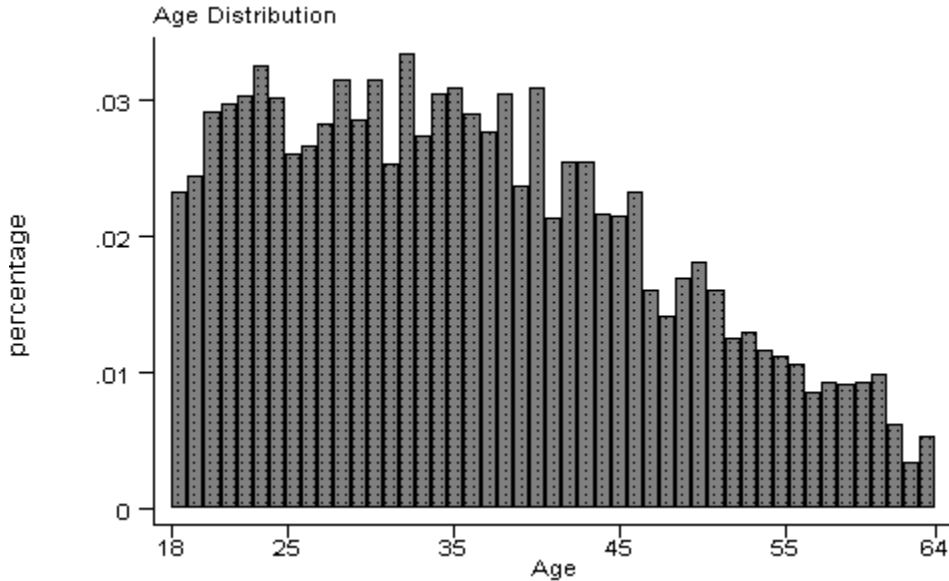
```
. graph age, bin(47) xlabel(18,25,35,45,55,64) ylabel(0, .01, .02, .03)
```



You might want to add titles to the graph. You can add titles on all four sides of the graph: Bottom, Left, Top, and Right. On each side you can add up to 2 titles. Suppose you want a top title (Age Distribution), a bottom title (Age), and a left title (percentage).  
Type:

```
. graph age, bin(47) xlabel(18,25,35,45,55,64) ylabel(0, .01, .02, .03)
> t1(Age Distribution) b2(Age) l1(percentage)
```

(notice that I typed the command on one long line: Stata divided it into two lines in the output)



**t1 (...)** stands for “title # 1 on TOP”; **b2 (...)** for “title # 2 – smallest – on BOTTOM”, etc.

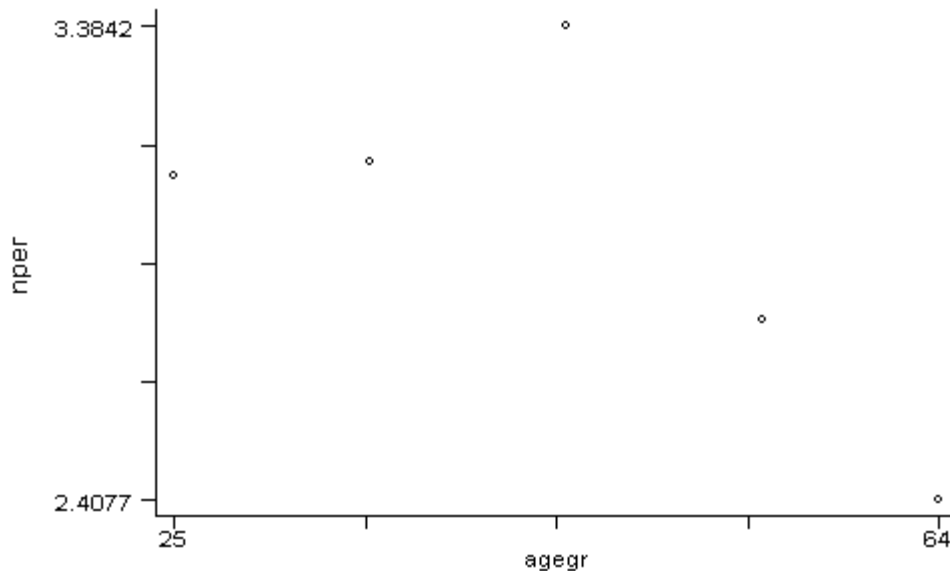
You can also plot scatterplot graphs. Suppose you have the average number of persons living in the household for each age group as defined before (you can build this variable as follows:

```
. egen nper=mean(fpersons), by(agegr)
```

the variable is called “nper”). You might want to plot the average number of persons in the household on one axis (say vertical) and the age group on the other (say horizontal).

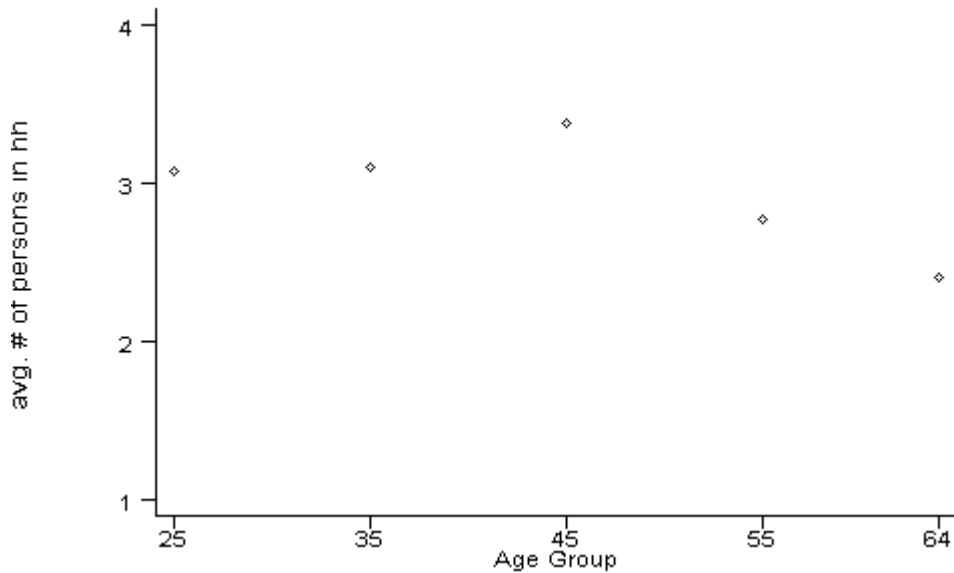
Type:

```
. graph nper agegr
```



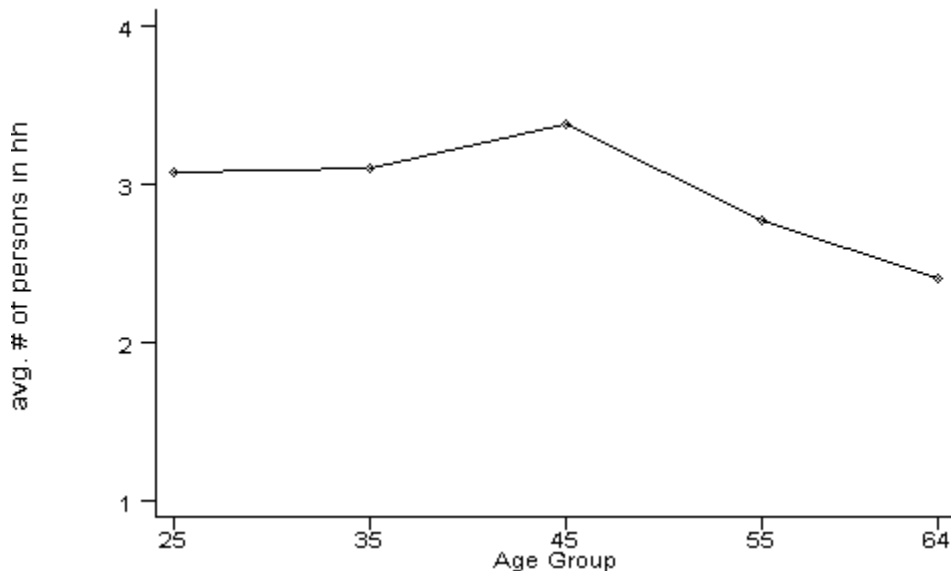
You could change the symbol for each point in the graph, and choose from various options: a square, a diamond, a triangle, a dot, a small circle, a large circle. If we want a diamond, (and we rescale the axis) we use option **s(d)** (“d” for “diamond”):

```
. graph nper agegr, xlabel(25,35,45,55,64) ylabel(1,2,3,4) b2(Age  
> Group) l1(avg. # of persons in hh) s(d)
```



We might also want to connect the points in the graph: use option **c(1)**

```
. graph nper agegr, xlabel(25,35,45,55,64) ylabel(1,2,3,4) b2(Age  
> Group) l1(avg. # of persons in hh) s(d) c(1)
```



In order to save a graph, go to the *File* menu and select *Save Graph*. You will save the graph, by default, in format *.gph*. You cannot import this format into Word. So you might want to save it as a Windows Metafile (*.wmf*) file that you can import into Word.

You can also copy the file to the Windows memory by going to the *Edit* menu and selecting *Copy Graph*. Then you can immediately paste it in the Word document that you are working in simply by clicking the “Paste” button in Word.

If you want to print your graph, in the *File* menu select *Print graph*.

If you want to see a graph that was saved in a file (say in C:\WINDOWS\Desktop\508\g1.gph), just type:

```
. graph using c:\WINDOWS\Desktop\508\g1
```

## .do files

In order to write a .do file, open an editor in Windows. You can use the Stata **Do-File Editor** (just click its button), or you can use any other text editor (e.g. Notepad, WordPad, Word, etc.). Always keep both Stata and the Editor open, because you will switch back and forth.

Every .do file is nothing more than a list of Stata commands. You write the commands in the Stata Editor, save the file and then run the .do file in Stata. Notice that if you don't use the Stata Editor you will have to specify that the file has to be saved as a .do file (in the *Save as type...* option select *All files (\*.\*)* and then enter the extension .do in the *File name* field).

How do you run the .do file?

If you're using the *Do-File Editor*, once you have changed your file you can simply click on the *Do current file* button.

In general, select *Do...* from the *File* menu, and double click on the .do file you want to run. You could also enter the command:

```
. do myfile
```

in order to run "myfile.do".

Here is an example of a .do file:

---

```
capture log close
log using test2.log, replace
drop _all

*****
* This file produces graphs of log(hourly wage) *
* distribution by sex. CPS March 93 (subsample) *
*****
* Date: Feb 10, 2003

set more 1

use c:\WINDOWS\Desktop\508\cps93.dta, clear
gen lnw=log(earnhr)
sort sex
#delimit;
graph lnw, bin(50) by(sex) total normal
      l1(fraction) b1(ln hour wage);
#delimit cr;
log close
```

The first 3 lines do the following:

1. close any open .log file
2. open the .log file where the results will be saved
3. clear the memory

All the lines beginning with \* are considered by Stata comments and not commands.

“set more 1” tells Stata not to stop at the end of each screen and wait for the “More” command.

Then file C:\WINDOWS\Desktop\508\cps93.dta is open, variable “lnw” is generated, and data are sorted according to “sex”.

“#delimit;” and “#delimit cr;” are commands that tell Stata to consider, respectively, “;” and Enter (or Carriage Return, `cr`) as the end of every command. The default for Stata is to consider every line as a command (`cr` is the default). But you can switch to “;”.

At the end, test2.log is closed.

## .log files

You can open a log file by typing:

```
. log using mylog.log
```

and close it simply by typing

```
. log close
```

You can also perform the same operations by using the *Log* button in Stata.

You can add a .log file to an existing one. Suppose you want to add your results to an existing .log file called C:\WINDOWS\Desktop\508\oldlog.log. You just type:

```
. log using c:\WINDOWS\Desktop\508\oldlog.log, append
```

or click on the *Log* button and click on *Append to existing file* when prompted the Stata Log Options window.

If, on the other hand, you want to erase the content of “mylog.log” and rewrite it:

```
. log using mylog.log, replace
```

or click on the *Log* button and click on *Overwrite existing file* when prompted the Stata Log Options window.

When a .log file is open, you can bring it to the front window by clicking the *Bring Log Window to Front* button. Once a .log file is open, all the commands you give and all their output will be “recorded” in the file. You could suspend this recording by clicking the *Log* button and the *Suspend log file* option; you can then resume the recording by clicking the *Log* button again and choosing *Resume suspended log file*.

When you want to close an open .log file, you just have to click the *Log* button and choose *Close log file*, or you can simply type:

```
. log close
```

What are .log files? They are text files (the same as files in format .txt) where you store Stata commands and their output. You can import them in Word documents, or in any word processor.

When you work interactively you can save your work (i.e., your commands and their output) on a .log file. Once you open it -- click on the *Open Log* button --, you can choose not to save all the commands and their output: if you want to stop saving and perform some commands that will not be saved in your log file, you can momentarily suspend it -- click again on the same button, now called *Close/Suspend Log*, and click on *Suspend log file*. You can then resume saving commands -- click the same button now called

*Resume/Close Log*, and chose *Resume log file* --, or you can close it -- as before, but chose *Close log file*. When a .log file is open, either suspended or active, you can bring the Log Window to the front by clicking on the *Bring Log Window to Front* button.

Example of a .log file:

---

```
use "C:\WINDOWS\Desktop\508\Cps93.dta", clear
(March 93 CPS discouraged extrac)
```

```
. sum
```

Variable	Obs	Mean	Std. Dev.	Min	Max
hhseq	8677	51098.76	18534.47	27999	73871
id	8677	41.76985	.9587741	41	52
lineno	8677	1.812954	1.092296	1	36
age	8677	36.07065	11.67076	18	64
marit	8677	3.348047	2.730244	1	7
sex	8677	1.516768	.4997475	1	2
educat	8677	39.32223	2.237435	31	46
race	8677	1.188429	.4925585	1	3
uslhrs	8677	35.98859	10.15854	1	99
earnhr	8677	9.321139	5.137567	.5	69.99
wstat	8677	2.523222	.9558349	2	5
lfsr	8677	1.03976	.1954068	1	2
clswk	8677	1.322807	.8534107	1	4
pid	8677	1.765933	.9545703	1	10
fid	8677	1.072836	.2900628	1	6
fpersons	8677	3.071108	1.517727	1	14
fownu6	8677	.2605739	.5751343	0	5
fownu18	8677	.8393454	1.107523	0	8
state	8677	51.8299	26.43896	11	95
hhid	8677	4.34e+11	3.28e+11	5.13e+07	1.00e+12
edtab3	8677	2.388498	.849794	1	4

```
. tab sex
```

Sex	Freq.	Percent	Cum.
male	4193	48.32	48.32
female	4484	51.68	100.00
Total	8677	100.00	

---

You can also open, suspend, resume and close .log files within a .do file, since all these operations are commands in Stata, and .do files are simple collections of commands in Stata.

Suppose I write the following .do file:

---

```

capture log close
log using c:\WINDOWS\Desktop\508\mylog, replace
use c:\WINDOWS\Desktop\508\Cps93.dta, clear
log off
describe
log on
summarize
tabulate sex
log close

```

---

This tells Stata do the following operations:

1. close any .log file that might be eventually open
2. open .log file `c:\WINDOWS\Desktop\508\mylog` overwriting any existing file with that name
3. load into Stata's memory the data file `c:\WINDOWS\Desktop\508\Cps93.dta` eventually clearing Stata's memory from any existing data set
4. suspend the recording of commands and output onto `c:\WINDOWS\Desktop\508\mylog`
5. describe the data in `c:\WINDOWS\Desktop\508\Cps93.dta`
6. resume the "recording" of data onto `c:\WINDOWS\Desktop\508\mylog`
7. summarize the data
8. give the distribution of the variable `sex`
9. close the .log file, i.e. stop the recording.

The output of this sequence of commands, as recorded on

`c:\WINDOWS\Desktop\508\mylog`, will be very similar to the one that we saw above:

---

```

. use c:\WINDOWS\Desktop\508\Cps93.dta, clear
(March 93 CPS discouraged extrac)

```

```

. log off

```

```

. summarize

```

Variable	Obs	Mean	Std. Dev.	Min	Max
hhseq	8677	51098.76	18534.47	27999	73871
id	8677	41.76985	.9587741	41	52
lineno	8677	1.812954	1.092296	1	36
age	8677	36.07065	11.67076	18	64
marit	8677	3.348047	2.730244	1	7
sex	8677	1.516768	.4997475	1	2
educat	8677	39.32223	2.237435	31	46
race	8677	1.188429	.4925585	1	3
uslhrs	8677	35.98859	10.15854	1	99
earnhr	8677	9.321139	5.137567	.5	69.99
wstat	8677	2.523222	.9558349	2	5
lfsr	8677	1.03976	.1954068	1	2

```

    clswk |      8677      1.322807      .8534107           1           4
      pid |      8677      1.765933      .9545703           1          10
      fid |      8677      1.072836      .2900628           1           6
fpersons |      8677      3.071108      1.517727           1          14
  fownu6 |      8677      .2605739      .5751343           0           5
  fownu18 |      8677      .8393454      1.107523           0           8
    state |      8677      51.8299      26.43896           11          95
    hhid |      8677      4.34e+11      3.28e+11      5.13e+07      1.00e+12
  edtab3 |      8677      2.388498      .849794           1           4

```

```
. tabulate sex
```

Sex	Freq.	Percent	Cum.
male	4193	48.32	48.32
female	4484	51.68	100.00
Total	8677	100.00	

```
. log close
```

---

It is thus very helpful to store your commands and your output in output files (.log files) that can be later inserted in other documents as tables.

## Some other useful commands

- **Confidence intervals:**

If you want to compute confidence intervals for a mean, you can do this using the command `ci`. Suppose you assume that the variable `lnw` is distributed according to a **normal** in the population, and you want to compute a confidence interval for its mean. You can do that by typing:

```
. ci lnw
```

Variable	Obs	Mean	Std. Err.	[95% Conf. Interval]	
lnw	8677	2.109741	.0052123	2.099524	2.119959

If you want to change the level of confidence, use the option `level()`

```
. ci lnw, level(99)
```

Variable	Obs	Mean	Std. Err.	[99% Conf. Interval]	
lnw	8677	2.109741	.0052123	2.096312	2.12317

If you want to compute the confidence interval for the mean of a population that is distributed according to a **binomial**, use the option `binomial`. Suppose you have a variable `male` that is equal to 1 when the person is a male and 0 when the person is a female. You want a 93% confidence interval for the proportion of males in the population:

```
. ci male, level(93) binomial
```

Variable	Obs	Mean	Std. Err.	-- Binomial Exact -- [93% Conf. Interval]	
male	8677	.4832315	.0053646	.4734598	.4930132

- **Correlation and covariance**

If you want to get the correlation matrix for two or more variables, the command you should use is `correlate` (shortcut `corr`). Suppose you're interested in the correlation between `lnw` and `age`: you type

```
. corr age lnw  
(obs=8677)
```

	age	lnw
age	1.0000	
lnw	0.2778	1.0000

If you're interested in the covariance matrix, you use the `covariance` (shortcut `cov`) option:

```
. corr age lnw, cov
(obs=8677)

      |      age      lnw
-----+-----
age |  136.207
lnw |   1.57429   .235741
```

On the main diagonal you have the sample variances of the variables, and in the lower triangle of the matrix you have the covariances.

You can also get the information about the means and minimum and maximum of each variable, using the option `means`:

```
. corr age lnw, cov means
(obs=8677)
Variable |      Mean      Std. Dev.      Min      Max
-----+-----
age |   36.07065   11.67076      18      64
lnw |    2.109741   .4855317   -.6931472   4.248353

      |      age      lnw
-----+-----
age |  136.207
lnw |   1.57429   .235741
```

If you are interested in the significance level of the correlation coefficients, you can use the command `pwcorr` with option `sig`:

```
. pwcorr age lnw, sig

      |      age      lnw
-----+-----
age |   1.0000
   |
lnw |   0.2778   1.0000
   |   0.0000
```

## Regressions

The basic command for linear regressions in Stata is **regress** (shortcut: **reg**). The first variable after the command is always the dependent (or left hand side, or *Y*) variable, and all the variables that follow are the independent ones (or right hand side, or *X*'s). By default Stata includes a constant to be estimated in every linear regression (you can estimate a regression without the constant by adding the option **noconstant**).

For example, using our data set from the CPS March 1993, let's do the following:

```
. gen lnw = ln(earnhr)
. gen male = (sex == 1)
```

We generated two variables: the natural logarithm of hourly wages (**lnw**) and a **dummy variable** for males (**male**)

We can run a linear regression of the logarithm of wages on age and a dummy variable for males (for an alternative way of running this regression without creating the dummy variable **male**, see the section on xi expansion below):

```
. reg lnw age male
```

Source	SS	df	MS	Number of obs =	8677
Model	237.041967	2	118.520983	F( 2, 8674) =	568.53
Residual	1808.24686	8674	.208467473	Prob > F =	0.0000
Total	2045.28882	8676	.235740989	R-squared =	0.1159
				Adj R-squared =	0.1157
				Root MSE =	.45658

lnw	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]
age	.01197	.0004205	28.463	0.000	.0111457 .0127944
male	.1913957	.009821	19.488	0.000	.1721441 .2106473
_cons	1.585486	.016849	94.100	0.000	1.552458 1.618515

Interpret all the elements in the table.

In order to construct the predicted value for each observation, you have to create a new variable (that I will call **lnwhat**). The command that you use in Stata to do so is:

```
. predict lnwhat
(option xb assumed; fitted values)
```

Notice that **predict** creates this new variable using the coefficients of the last regression that was estimated. If you estimated different equations, and you want to build the estimated values for each one of them, you have to do so before you go to the next equation. Otherwise all the information would be lost.

You can also predict the estimated residual term for each observation. In order to do so you have to create a new variable, that I will call **res**:

```
. predict res, residual
```

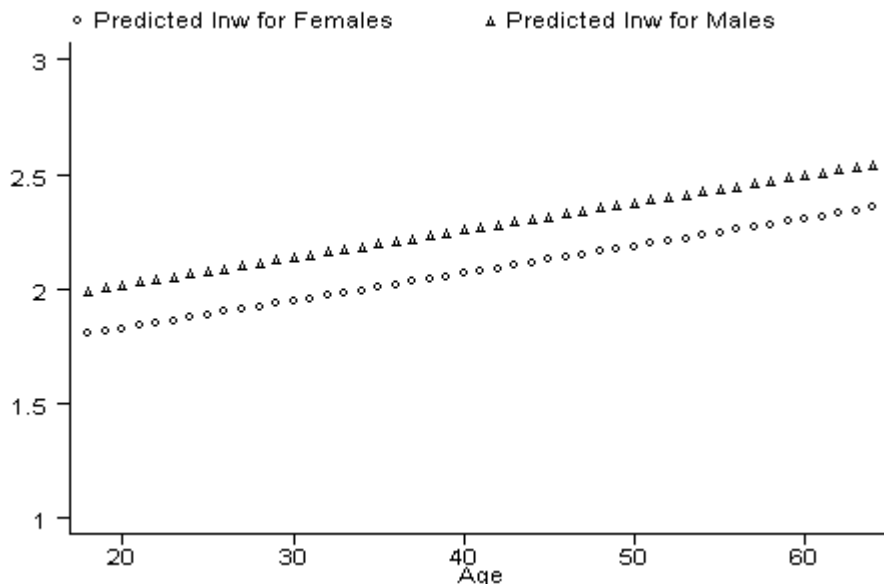
You can then plot your predicted values in a scatter plot diagram together with the original values. notice that in this case we are estimating an equation where there are two independent variables. One is a continuous variable, and the other is a dummy variable. So we are estimating two lines that have the same "slope" but two different "intercepts". Each line characterizes a different group (males and females), so we might want to call the predicted values for the two groups with different names:

```
. gen lnwm=lnwhat if male==1
(4484 missing values generated)
. label var lnwf "Predicted lnw for Females"

. gen lnwf=lnwhat if male==0
(4193 missing values generated)
. label var lnwm "Predicted lnw for Males"
```

If we now plot the predicted values for the two groups we get:

```
. graph lnwf lnwm age, ylabel(1,1.5,2,2.5,3) xlabel(20,30,40,50,60)
(see notes on graphs)
```



Here you can see how the dummy variable changes the intercept for the two predicted lines.

You can also think that there might be different slopes for the two groups. In this case you want to create a variable that is called an **interaction** variable: the product of **age** times **male**. I will call it **age\_m**:

```
. gen age_m=age*male
```

Then you can estimate the equation with the interaction term (for an alternative way of running regressions with interaction terms when one of the interacted variables is categorical, see section on xi expansion below):

```
. reg lnw age male age_m
```

Source	SS	df	MS	Number of obs = 8677		
Model	251.574222	3	83.8580741	F( 3, 8673)	=	405.47
Residual	1793.7146	8673	.206815935	Prob > F	=	0.0000
				R-squared	=	0.1230
				Adj R-squared	=	0.1227
				Root MSE	=	.45477

lnw	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
age	.0085482	.0005849	14.615	0.000	.0074017	.0096947
male	-.0617471	.0317437	-1.945	0.052	-.1239723	.0004781
age_m	.0070248	.000838	8.383	0.000	.005382	.0086675
_cons	1.710854	.0224793	76.108	0.000	1.66679	1.754919

Re-computing predicted values, we have to create two new variables for this second equation in order to differentiate them from the previous one:

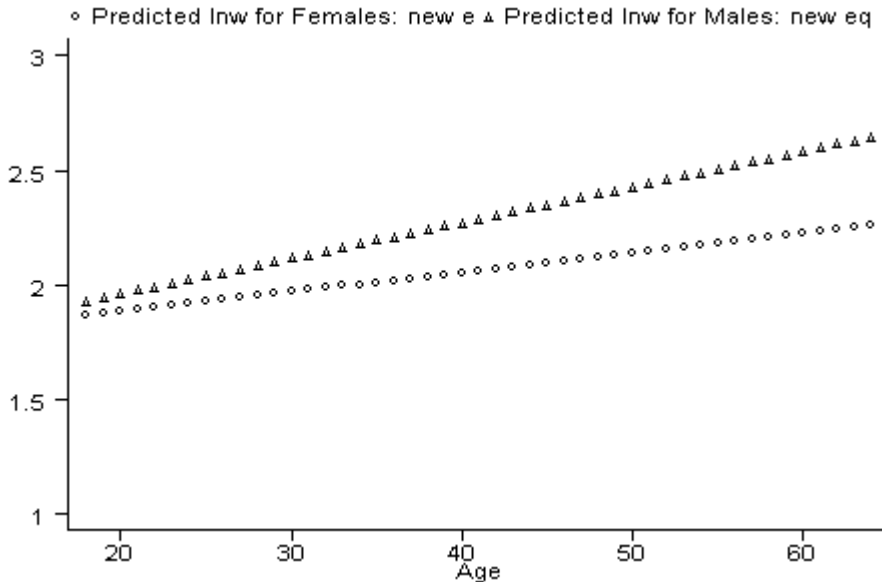
```
. predict lnwf_new if male==0
(option xb assumed; fitted values)
(4193 missing values generated)
. predict lnwm_new if male==1
(option xb assumed; fitted values)
(4484 missing values generated)

. label var lnwm_new "Predicted lnw for Males: new eq"
. label var lnwf_new "Predicted lnw for Females: new eq"
```

(Notice the use of the "if" condition in the prediction).

And we can graph the result:

```
. graph lnwf_new lnwm_new age, ylabel(1,1.5,2,2.5,3)
xlabel(20,30,40,50,60)
```



Here you can see the different slopes and intercepts for both groups.

If you want to perform tests of hypotheses on the coefficients in the equation, you can do so using the command **test**.

For example, suppose you want to test the following hypothesis:

$$H_0: \beta_{age} = 0.01 \quad H_1: \beta_{age} \neq 0.01$$

You just type:

```
. test age = = 0.01

( 1)  age = .01
      F( 1, 8673) =    6.16
      Prob > F =    0.0131
```

(Interpret the result).

If you want to test a more complicated hypothesis:

$$H_0: \beta_{age} = \beta_{age\_m} \quad H_1: \beta_{age} \neq \beta_{age\_m}$$

you just type:

```
. test age==age_m

( 1)  age - age_m = 0.0
      F( 1, 8673) =    1.34
      Prob > F =    0.2466
```

If you want to test another hypothesis:

$$H_0: \beta_{age} = \beta_{age\_m} = 0 \quad H_1: \beta_{age} \neq 0 \text{ or } \beta_{age\_m} \neq 0$$

then you type:

```
. test age age_m

( 1)  age = 0.0
( 2)  age_m = 0.0
      F( 2, 8673) =  443.45
      Prob > F =    0.0000
```

## **xi expansions and interaction terms**

**xi** expands terms containing categorical variables into indicator (dummy) variable sets, and when used with a specific Stata command, executes the specific command using the expanded terms. If the variable attains  $k$  values, **xi** creates  $(k-1)$  dummy variables.

For example, suppose you want to create dummy variables for the variable **race** (remember, race has 3 categories: white, black and other). You could just type

```
. xi i.race
i.race      _Irace_1-3      (naturally coded; _Irace_1 omitted)
```

What did Stata do?. It created TWO dummy variables, automatically considering as omitted category the most frequent category (`race == 1`). In other words, it created one dummy variable for “blacks” (`race == 2`) and one dummy variable for “others” (`race == 3`). These two dummy variables were automatically named `_Irace_2` and `_Irace_3`. You can look at these variables by typing `des _Irace*`:

```
. des _Irace*
```

variable name	storage type	display format	value label	variable label
<code>_Irace_2</code>	byte	%8.0g		race==2
<code>_Irace_3</code>	byte	%8.0g		race==3

In order to change to omitted category, you can specify which category you want to leave out. Suppose you wanted to have “blacks” as the omitted category. You would have to specify:

```
. char race[omit] 2
```

before creating the dummy variables. If you were to perform `xi` now, the outcome would be:

```
. xi i.race
i.race      _Irace_1-3      (naturally coded; _Irace_2 omitted)

. des _Irace*
```

variable name	storage type	display format	value label	variable label
<code>_Irace_1</code>	byte	%8.0g		race==1
<code>_Irace_3</code>	byte	%8.0g		race==3

You can also change the prefix that Stata automatically assigns to the newly created variables (by default, the new variables are named `1oldvar`). Suppose you wanted to name `race_1` the dummy variable for “whites” and `race_3` the dummy variable for “others”. You would type:

```
. xi , prefix() i.race
i.race      race_1-3      (naturally coded; race_2 omitted)

. des race*
```

variable name	storage type	display format	value label	variable label
<code>race</code>	byte	%8.0g	racelab	Race
<code>race_1</code>	byte	%8.0g		race==1
<code>race_3</code>	byte	%8.0g		race==3

Now that we know how to create dummy variables using `xi`, it will be straight forward to apply `xi` to regressions.

Suppose you want to run a linear regression of the logarithm of wages on age, a dummy variable for males (as in the example above). You would give the following commands:

```
. char sex[omit] 2
. xi, prefix(): reg lnw age i.sex
i.sex          sex_1-2          (naturally coded; sex_2 omitted)
```

Source	SS	df	MS	Number of obs =	8677
Model	237.041967	2	118.520983	F( 2, 8674) =	568.53
Residual	1808.24686	8674	.208467473	Prob > F =	0.0000
				R-squared =	0.1159
				Adj R-squared =	0.1157
Total	2045.28882	8676	.235740989	Root MSE =	.45658

lnw	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]
age	.01197	.0004205	28.46	0.000	.0111457 .0127944
sex_1	.1913957	.009821	19.49	0.000	.1721441 .2106473
_cons	1.585486	.016849	94.10	0.000	1.552458 1.618515

Notice that the outcome is identical to what we got above (see page 26).

You can also “expand” more than one categorical variable at a time. Suppose you want to run a linear regression of the logarithm of wages on age, a dummy variable for males, a dummy variable for “black” and a dummy variable for “other”. You would type the following sequence of commands:

```
. char race[omit] 1
. char sex[omit] 2
. xi, prefix(): reg lnw age i.sex i.race
i.sex          sex_1-2          (naturally coded; sex_2 omitted)
i.race         race_1-3        (naturally coded; race_1 omitted)
```

Source	SS	df	MS	Number of obs =	8677
Model	241.886246	4	60.4715615	F( 4, 8672) =	290.79
Residual	1803.40258	8672	.207956939	Prob > F =	0.0000
				R-squared =	0.1183
				Adj R-squared =	0.1179
Total	2045.28882	8676	.235740989	Root MSE =	.45602

lnw	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]
age	.0120079	.0004201	28.58	0.000	.0111844 .0128314
sex_1	.1908842	.0098096	19.46	0.000	.171655 .2101134
race_2	-.0793157	.0164553	-4.82	0.000	-.111572 -.0470595
race_3	-.0023435	.023719	-0.10	0.921	-.048838 .0441513
_cons	1.592306	.0169241	94.09	0.000	1.559131 1.625481

Notice that Stata specifies which categories have been omitted. How should you interpret the results? If you take two individuals with the same gender and age, one black and the other white, the former will earn (on average) 7.9% less than the latter. At the same time, men earn 19.1% more than women, controlling for race and age.

`xi` also allows you to include interaction terms.

Suppose you want to replicate the regression of page 29. In this case you would type:

```
. xi, prefix(): reg lnw i.sex*age
```

i.sex		sex_1-2		(naturally coded; sex_2 omitted)	
i.sex*age		sexXage_#		(coded as above)	
Source	SS	df	MS	Number of obs	= 8677
Model	251.574222	3	83.8580741	F( 3, 8673)	= 405.47
Residual	1793.7146	8673	.206815935	Prob > F	= 0.0000
Total	2045.28882	8676	.235740989	R-squared	= 0.1230
				Adj R-squared	= 0.1227
				Root MSE	= .45477

lnw	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
sex_1	-.0617471	.0317437	-1.95	0.052	-.1239723	.0004781
age	.0085482	.0005849	14.62	0.000	.0074017	.0096947
sexXage_1	.0070248	.000838	8.38	0.000	.005382	.0086675
_cons	1.710854	.0224793	76.11	0.000	1.66679	1.754919

Notice that the output is the same as the one on page 29. The variable that we called `age_m` has now been automatically created by Stata, and it's called `sexXage_1`.

You could also run a linear regression of the logarithm of wages on age, a dummy variable for males, a dummy variable for "black" and a dummy variable for "other", and interaction terms between sex and race.

```
. xi, prefix(): reg lnw age i.sex*i.race
```

i.sex		sex_1-2		(naturally coded; sex_2 omitted)	
i.race		race_1-3		(naturally coded; race_1 omitted)	
i.sex*i.race		sexXrac_#_#		(coded as above)	
Source	SS	df	MS	Number of obs	= 8677
Model	242.863583	6	40.4772639	F( 6, 8670)	= 194.70
Residual	1802.42524	8670	.207892185	Prob > F	= 0.0000
Total	2045.28882	8676	.235740989	R-squared	= 0.1187
				Adj R-squared	= 0.1181
				Root MSE	= .45595

lnw	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
-----	-------	-----------	---	------	----------------------	--

age		.0120114	.0004201	28.59	0.000	.011188	.0128349
sex_1		.1995337	.0105954	18.83	0.000	.1787643	.220303
race_2		-.0527068	.0225456	-2.34	0.019	-.0969016	-.008512
race_3		.0303354	.0328415	0.92	0.356	-.0340418	.0947125
sexXrac_1_2		-.0567876	.0329715	-1.72	0.085	-.1214195	.0078444
sexXrac_1_3		-.0682125	.0474762	-1.44	0.151	-.1612772	.0248521
_cons		1.587979	.017044	93.17	0.000	1.554569	1.621389

---

As you can see Stata automatically created two extra variables, the interaction terms **sexXrac\_1\_2** and **sexXrac\_1\_3**. The first is the product of **sex\_1** by **race\_2**, and the second is the product of **sex\_1** by **race\_3**.