An Introductory Guide to Stata

Scott L. Minkoff
Assistant Professor
Department of Political Science
SUNY New Paltz
minkoffs@newpaltz.edu
www.scottminkoff.com

Version 2
Updated: July 9, 2012

Table of Contents

About this Guide

This guide introduces you to the commands necessary to do basic statistical analysis in Stata. It emphasizes (and goes a bit beyond) the commands taught in my Designing Social Inquiry course but should be generally applicable to new users of Stata. The guide was written based on Stata 11 for a Mac.* Most (if not all) of the commands presented here are consistent across Mac and Windows versions Stata and compatible with all versions since Stata 9 (and in most cases, earlier versions as well).

The goal is that users of this guide will become proficient enough in both statistics and Stata that they will be able to move beyond the commands explained here. For example, this guide provides minimal support for more advanced regression techniques such as time-series analysis and regression for categorical dependent variables. However, before users of Stata can learn to do these more advanced techniques, they will need to have strong grasp of the commands reviewed in this guide.

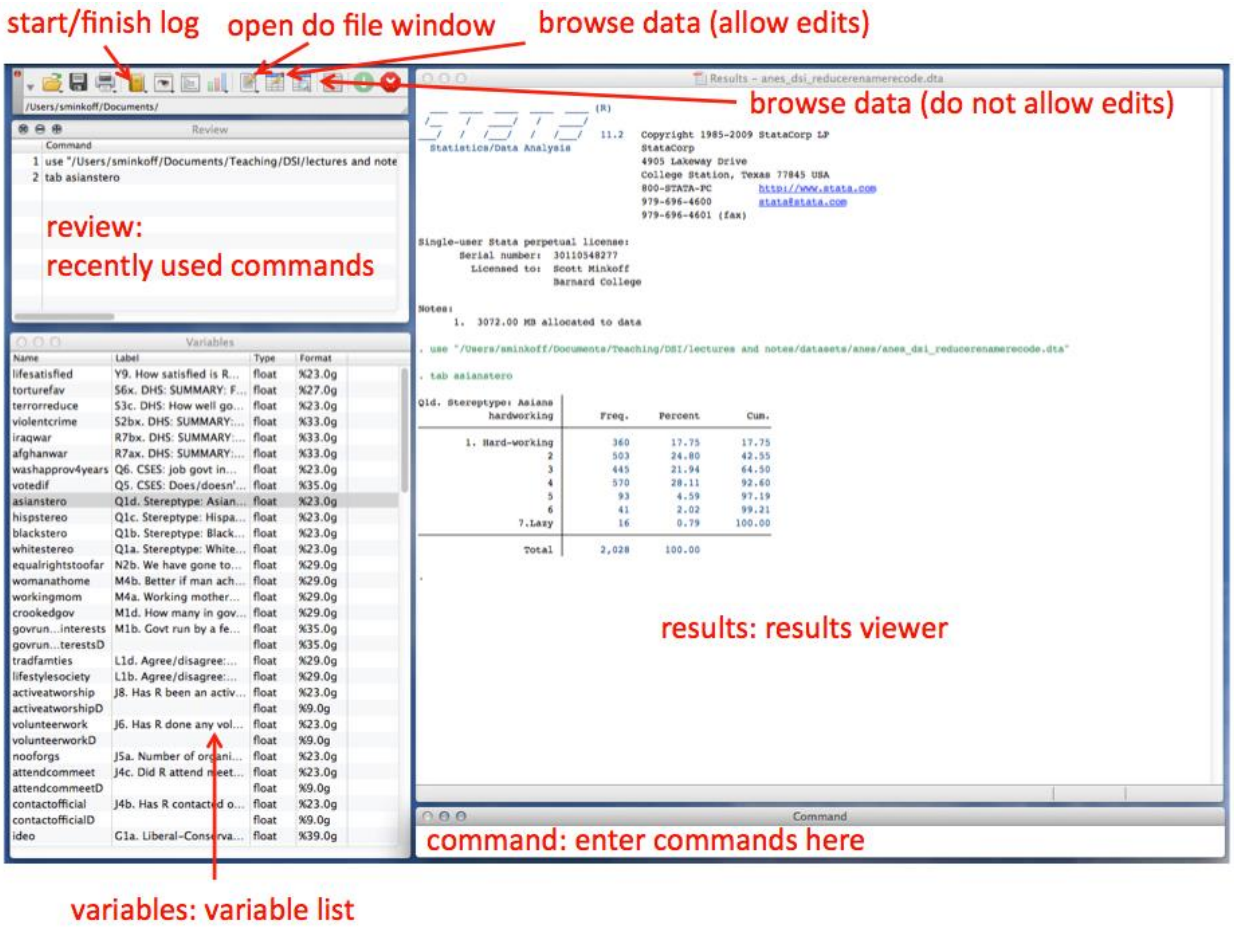
The guide uses different fonts in order to distinguish between what is explanation and what should actually be entered into Stata. Stata commands are written in `Lucinda Console font`.

*StataCorp. 2009. *Stata Statistical Software: Release 11*. College Station, TX: StataCorp LP.

Introduction to Stata

The Stata Interface

The Stata interface is slightly different based on which release of Stata you are using and which operating system you are on. However, the basic windows in Stata are pretty consistent so Stata should look roughly like this:



Stata has 4 primary windows:

Review Window: A list of commands that you recently used in the order in which you used them. Commands that failed to execute are listed in red.

Variables Window: A list of all the variables in the dataset.

Command Window: The window where you enter your commands.

Results: The window that displays the results.

Additionally, some functions are easy to access from the top menu:

Start/Finish Log: Starts and finishes a log of everything that is occurring in the results window.

Open Do-File Window: Allows use to open a do-file window.

Browse Data (allow edits): Opens the data in spreadsheet form and allows you to make changes to the data.

Browse Data (do not allow edits): Opens the data in spreadsheet form but prevents you from editing the data.

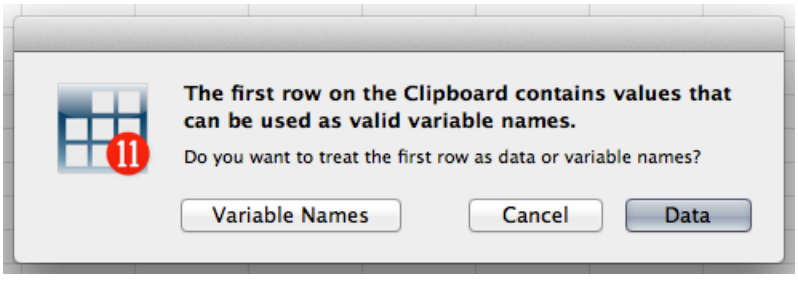
Loading Data Into Stata

Stata uses files with the extension "dta". If your data is already in a dta format, then you can:

File → Open → Select your dta file

If your file is not in "dta" format, the best thing to do is open it in Excel and then copy and paste it into Stata. Prior to doing this, make sure that the first row of your Excel spreadsheet has the variable names.

Open the data in Excel
Highlight all the data (including the variable names)
Open Stata and click the "Browse Data with Edits Icon"
Click on the upper-left most cell
Paste the data (ctrl-v)
When prompted, tell Stata to treat the first row as variable names



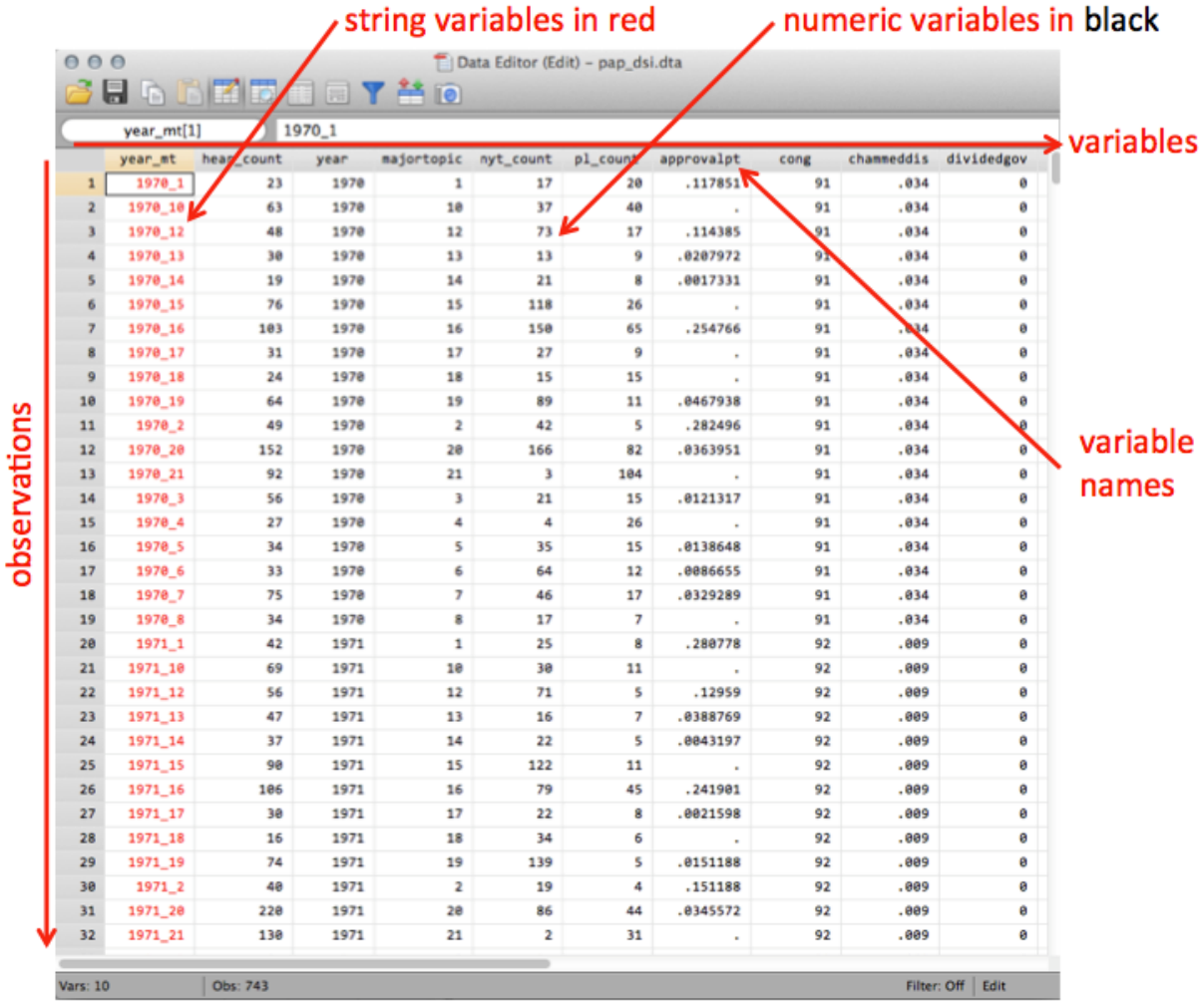
Save your data as a dta file

There are two more options. First, for users of Stata 12, Excel files can be opened directly into Stata. Make sure to save it as a dta file after you open it. Second, for more advanced users,

Stata has an "insheet" command that allows users to import text files (csv) by providing the file path.

Viewing Your Data in Stata

In Stata, your data in stored (and can be viewed) as a spreadsheet—similar to one you might view or make in Excel. While this spreadsheet is not nearly as customizable as the spreadsheets in Excel, it does allow you to get a sense of what your data look like.



	year_mt	hear_count	year	majortopic	nyt_count	pl_count	approvalpt	cong	chamneddis	dividedgov
1	1970_1	23	1970	1	17	20	.117851	91	.034	0
2	1970_10	63	1970	10	37	40	.	91	.034	0
3	1970_12	48	1970	12	73	17	.114385	91	.034	0
4	1970_13	30	1970	13	13	9	.0207972	91	.034	0
5	1970_14	19	1970	14	21	8	.0017331	91	.034	0
6	1970_15	76	1970	15	118	26	.	91	.034	0
7	1970_16	103	1970	16	150	65	.254766	91	.034	0
8	1970_17	31	1970	17	27	9	.	91	.034	0
9	1970_18	24	1970	18	15	15	.	91	.034	0
10	1970_19	64	1970	19	89	11	.0467938	91	.034	0
11	1970_2	49	1970	2	42	5	.282496	91	.034	0
12	1970_20	152	1970	20	166	82	.0363951	91	.034	0
13	1970_21	92	1970	21	3	104	.	91	.034	0
14	1970_3	56	1970	3	21	15	.0121317	91	.034	0
15	1970_4	27	1970	4	4	26	.	91	.034	0
16	1970_5	34	1970	5	35	15	.0138648	91	.034	0
17	1970_6	33	1970	6	64	12	.0086655	91	.034	0
18	1970_7	75	1970	7	46	17	.0329289	91	.034	0
19	1970_8	34	1970	8	17	7	.	91	.034	0
20	1971_1	42	1971	1	25	8	.280778	92	.009	0
21	1971_10	69	1971	10	30	11	.	92	.009	0
22	1971_12	56	1971	12	71	5	.12959	92	.009	0
23	1971_13	47	1971	13	16	7	.0388769	92	.009	0
24	1971_14	37	1971	14	22	5	.0043197	92	.009	0
25	1971_15	90	1971	15	122	11	.	92	.009	0
26	1971_16	106	1971	16	79	45	.241901	92	.009	0
27	1971_17	30	1971	17	22	8	.0021598	92	.009	0
28	1971_18	16	1971	18	34	6	.	92	.009	0
29	1971_19	74	1971	19	139	5	.0151188	92	.009	0
30	1971_2	40	1971	2	19	4	.151188	92	.009	0
31	1971_20	220	1971	20	86	44	.0345572	92	.009	0
32	1971_21	130	1971	21	2	31	.	92	.009	0

Stata Help

Stata offers pretty good help within the program. When you are confused about how to use a command, you can type: `help` in the command window and it will bring up the Stata help guide. You can also be more specific, for example:

```
help histogram
help gen
help tab
```

UCLA's Academic Technology Services website is also an excellent resource for Stata users. In addition to reviewing Stata commands, the site offers examples of analyses with annotated outputs.

Homepage: http://www.ats.ucla.edu/stat/stata/
Data Analysis Examples: http://www.ats.ucla.edu/stat/dae/
Annotated Outpu: http://www.ats.ucla.edu/stat/AnnotatedOutput/

Logs

Stata logs allow you to easily keep track of everything you have done (all the commands you have entered and results you have produced). If you want to keep a log, click the log icon before you get started working. The log will run in the background recording your work. When you are done working, select the log icon again and tell Stata you want to close the log. When you want to review the log, navigate to the file and select it—it should open up in Stata.

Do-Files

Do-Files are the best way to make keep track of what you have done so that you can do it again another time. Rather than actually modifying your data permanently, you can put all your commands in the Do-File and run them at once. Once you have a grasp on Stata commands, I encourage you to revisit (and experiment) with Do-Files.

Types of Variables

There are two basic types of variables in Stata:

Numeric Variables: Variables that take a numerical value. Note that when the values of numeric variables are labeled in Stata, then the label appears in the data viewer rather than the number. Missing numeric data in Stata is recorded as a period (.).

String Variables: Variables that are non-numeric (primarily letters and symbols). Note that string variables can contain numbers but in this form Stata cannot process the variable for statistical analysis.

Commands in Stata

Commands in Stata generally take the following form:

```
command variable-list
```

The "command" tells Stata what it is going to be doing (making a table, making a graph, computing a statistic, running a regression, etc.). Occasionally multiple commands are needed. In

these situations you will have a general command (e.g. telling Stata to make a graph) and then sub-command (e.g. telling Stata which kind of graph to make). The "variable-list" tells Stata which variables to use for the action or analysis. Many commands in Stata allow for options. Options are added on to the end of a command following a comma.

```
command variable-list, option
```

Basic Stata Operators

These operators will come in handy with various commands. In particular, you will find them useful when manipulating variables.

- `+` addition
- `-` subtraction
- `*` multiplication
- `/` division
- `^` power (exponent)
- `-` negative
- `&` and
- `|` or
- `~` not
- `!` not
- `>` greater than
- `<` less than
- `>=` gerater than or equal to
- `<=` less than or equal to
- `==` equal to
- `!=` not equal to
- `~=` not equal to

Basic Summary Commands

Summarize a variable

Reports the number of non-missing observations, the mean, standard deviation, minimum, and maximum for the specified variable (in this case, var1). More than one variable can be included.

```
sum var1
```

Variable	Obs	Mean	Std. Dev.	Min	Max
var1	2028	2.861933	1.308645	1	7

Variable	Obs	Mean	Std. Dev.	Min	Max
var3	2028	2.861933	1.308645	1	7

Table a variable

Get the frequency, percentage, and cumulative percentage for ordinal and categorical variables. Stata will table continuous variable so long as the variable does not take too many values (the limit depends on which version of Stata you are using).

```
tab var1
```

var1	Freq.	Percent	Cum.
1	360	17.75	17.75
2	503	24.80	42.55
3	445	21.94	64.50
4	570	28.11	92.60
5	93	4.59	97.19
6	41	2.02	99.21
7	16	0.79	100.00
Total	2,028	100.00	

Get a specific statistic

Mean:

```
tabstat var1, stat(mean)
```

variable	mean
var1	2.861933

Median: To get the median, you actually ask Stata for the value of the 50th percentile.

```
tabstat var1, stat(p50)
```

variable	p50
var1	3

You can also ask for more than one statistic at a time:

Max and Minimum:

```
tabstat var1, stat(min max)
```

variable	max	min
var1	7	1

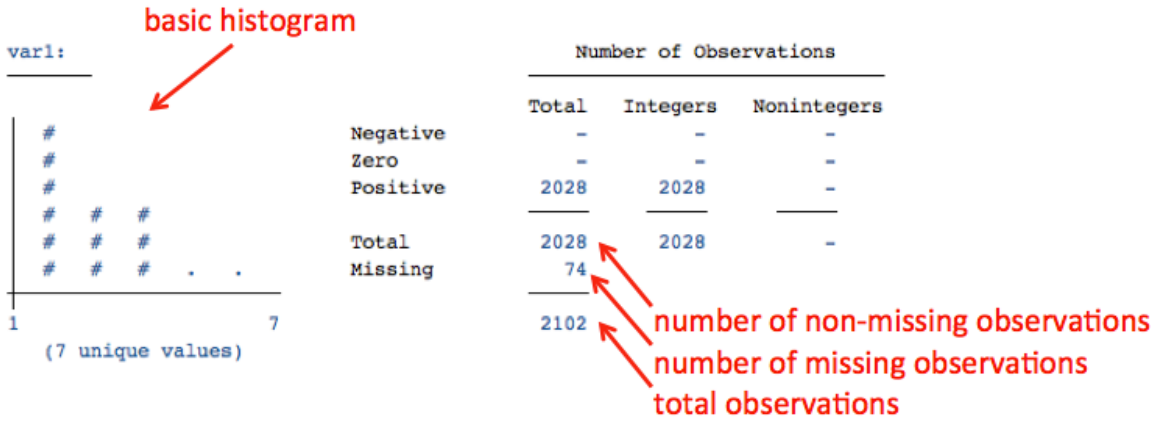
Standard Deviation and Inter-Quartile Range (IQR):

```
tabstat var1, stat(sd iqr)
```

variable	sd	iqr
var1	1.308645	2

Inspect a variable

```
inspect var1
```

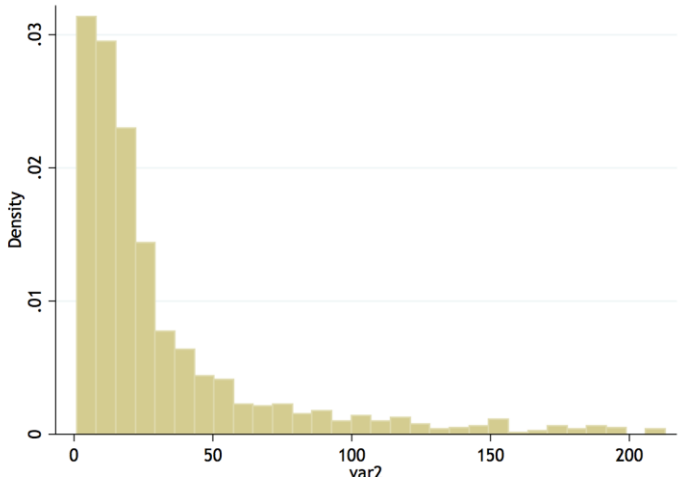


Descriptive Graphs

Histograms

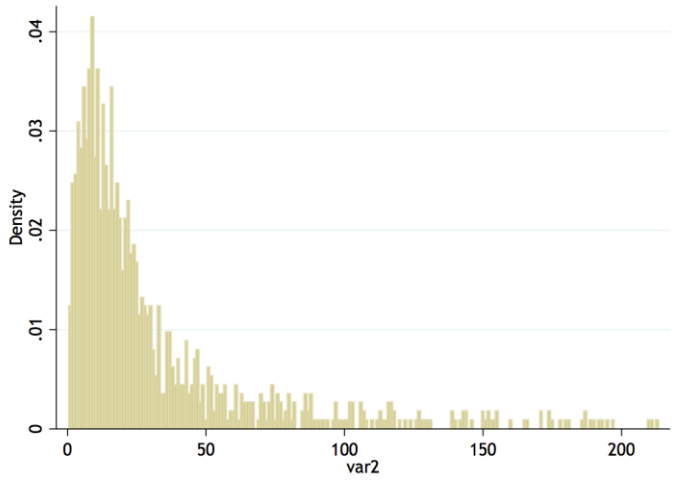
Standard Histogram:

```
histogram var2
```



Discrete Histogram (produces a separate bar for each possible value):

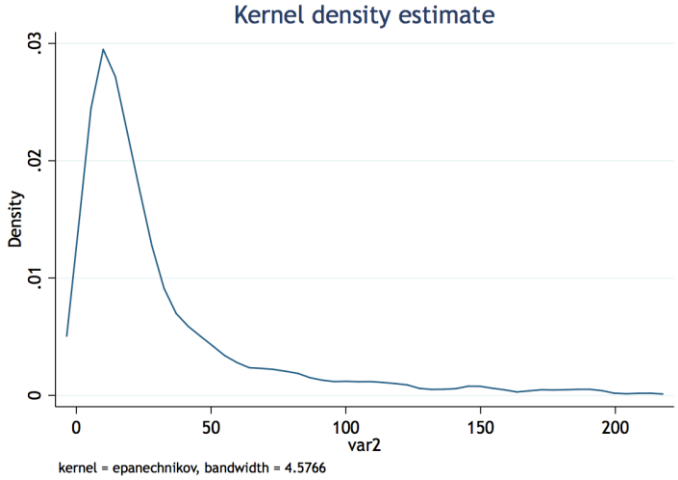
```
histogram var2, d
```



Other Descriptive Graphs: Kernel Density, Box and Whisker

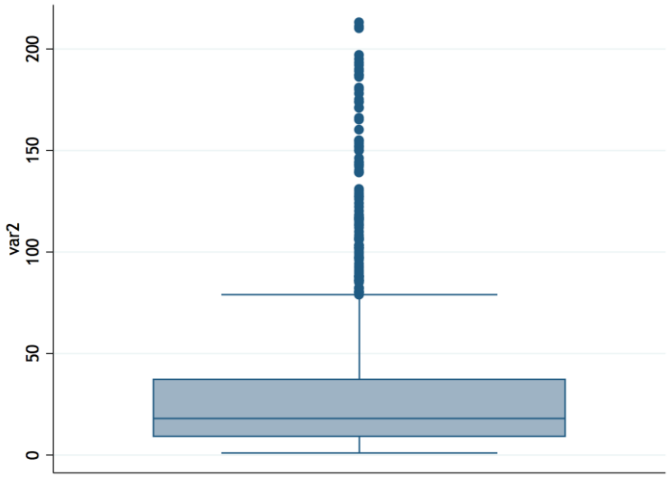
Kernel Density Graph:

```
kdensity var2
```



Box and Whisker Graph:

```
graph box var2
```



Note that some graph commands require you to put "graph" at the beginning of the command. histogram and kdensity do not require this, box does.

Generating and Manipulating Variables in Stata

Naming Variables in Stata

Variable names can use letters, numbers, and underscores. They cannot start with a number and cannot include spaces. Variable names can be up to 32 characters long. Ideally, you want to keep you variable names as short and descriptive as you can. Short variable names are easier to type and descriptive variable names are easier to identify. As you manipulate variables, you can name them to reflect the manipulation. For example, you may have variable called "unemp09" that is the unemployment rate for each observation in 2009. If you create a new variable that is the log of "unemp09" then you could name it "unemp09_log".

The Basic "gen" Command

The **gen** allows you to create new variables based on other variables. Most often, these new variables will be based on other variables in the dataset.

Generate a variable equal to another variable already in the dataset. In the following example, Stata will generate a new variable named "var3" that is exactly the same as var1.

```
gen var3 = var1
```

Generate a variable that takes a specific value. In the following example, Stata will generate a new variable named "var3" in which all observations take the value 1.

```
gen var3=1
```

Generate a variable where some amount is added to all the values of another variable. In the following example, Stata will generate a new variable named "var3" that takes the values of each observation in var2 and adds 1 to them.

```
gen var3 = var2 + 1
```

var2	var3
1	2
4	5
2	3
4	5
0	1
-3	-2
8	9

Add, Subtract, Multiply, and Divide Variables

Generate a variable that is the sum of two other variables:

```
gen var3 = var1 + var2
```

var1	var2	var3
1	3	4
1	1	2
0	0	0
3	0	3
1	2	3
4	3	7
4	0	4
3	1	4

Use the same procedure to subtract (-), multiply (*), and divide (/) variables.

Log Variables

Create a new variable that is the natural log of another variable:

```
gen var3 = ln(var2)
```

Exponentiate Variables

Create a new variable that is the square, cube, etc. of another variable:

```
gen var3 = var2^2
```

Generate Special Variables

Stata has lots of other ways to develop variables, many of which fall under the **egen** command. To learn more type: **help egen**. Here is one example:

Generate a variable that is the average of several other variables:

```
egen var4 = rowmean(var1 var2 var3)
```

var1	var2	var3	var4
1	3	4	2.666
1	1	2	1.333
0	0	0	0
3	0	3	2
1	2	3	2
4	3	7	4.666
4	0	4	2.666

Recode Variables

Recoding variables involves changing a specific value of a variable to another specific value. Often it is a good idea to generate a new variable before you recode it. For example, before recoding the values on var1, generate a new variable called var1b (gen var1b = var1) and then work with the new variable.

Recode all 1s to be 0s:

```
recode var1 (1=0)
```

Recode all 1s to be 0s AND 2s to be 1s:

```
recode var1 (1=0) (2=1)
```

Recode all 9s to be "missing":

```
recode var1 (9=.)
```

Recode all values between 3 and 6 to be 4.5:

```
recode var1 (3/6=4.5)
```

Replace Values of Variables

Replace can be used to do some of the same things as recode but has additional capabilities. In particular, the replace command allows you to recode variables based on the values they take in other variables. In the commands below, you will note the use of the double equals sign (==). In Stata, the == is used in conditional situations (frequently following an "if"): values in a variable are being replaced with another value based on a condition. In the first example, the condition is when var1 takes the value 1 in var1. In the second example, the condition is when var1 takes the value 3 in another variable (var3). Again, it is often a good idea to generate a new variable before you replace values. For example, before replacing the values on var1, generate a new variable called var1b (gen var1b = var1) and then work with the new variable.

Replace all 1s to be 0s:

```
replace var1=0 if var1==1
```

var1 (before)	var1 (after)
1	0
1	0
0	0
3	3
1	0
4	4

Replace all 1s in var1 to be 0s if they are 3s in var2: `replace var1=0 if var2==3`

var1 (before)	var3	var1 (after)
1	3	0
1	1	1
0	0	0
3	0	3
1	2	1
4	3	0
4	0	4

Rename a Variable

Rename var1 to be named "unemp":

```
rename var1 unemp
```

Labeling Variables

Sometimes it is nice to have a brief description of a variable. This description will appear in the variable viewer so that you can more easily identify it.

Give the variable var1 the label "unemployment rate":

```
label var var1 “unemployment rate”
```

Note the first “var” is part of the command. If the variable being labeled was “unemp” the command would look like:

```
label var unemp “unemployment rate”
```

Label Variable Values

When working with categorical variables it can be helpful to label the actual values of the variable.

Let’s say we have a variable for political ideology called “ideo” where:

1 = Extremely liberal
2 = Liberal
3 = Slightly liberal
4 = Moderate
5 = Slightly conservative
6 = Conservative
7 = Extremely conservative

When the values of the ideology variable are unlabeled, the table for the variable (ideo) will look like:

ideo	Freq.	Percent	Cum.
1	64	4.03	4.03
2	196	12.35	16.38
3	197	12.41	28.80
4	497	31.32	60.11
5	243	15.31	75.43
6	306	19.28	94.71
7	84	5.29	100.00
Total	1,587	100.00	

When the values of ido are labeled, the table can look like:

political ideology	Freq.	Percent	Cum.
1. Extremely liberal	64	4.03	4.03
2. liberal	196	12.35	16.38
3. Slightly liberal	197	12.41	28.80
"4.Moderate"	497	31.32	60.11
5. Slighly conservative	243	15.31	75.43
6. Conservative	306	19.28	94.71
7. Extremely conservative	84	5.29	100.00
Total	1,587	100.00	

To label values, you first must define the label:

```
label define ideolab 1 "1. Extremely liberal" 2 "2. liberal" 3 "3.
Slightly liberal" 4 “4.Moderate” 5 "5. Slighly conservative" 6 "6.
Conservative" 7 "Extremely conservative"
```

Then, you apply the label to the variable:

```
label values ideolab ideo
```

You can apply the defined values to as many variables as you want. Note that the label values command is finicky and little things can make the whole thing not work. Keep your labels as basic as possible to avoid problems.

CROSSTABS

Crosstabs are just an extension of the table command described above. Note that the first variable listed in the command (var3) runs vertically and the second variable listed in the command (var4) runs horizontally.

Basic crosstab where frequencies are reported:

```
tab var3 var4
```

number of observations that take a 1 for var3 and 0 for var4

var3	var4 0	1	Total
1	196	164	360
2	306	197	503
3	257	188	445
4	353	217	570
5	42	51	93
6	22	19	41
7	6	10	16
Total	1,182	846	2,028

row totals

column totals

Crosstab with frequencies and column percentages (`col`):

```
tab var3 var4, col
```

percentage of observations that take a 0 for var4 that also take 1 on var3

var3	var4 0	1	Total
1	196 16.58	164 19.39	360 17.75
2	306 25.89	197 23.29	503 24.80
3	257 21.74	188 22.22	445 21.94
4	353 29.86	217 25.65	570 28.11
5	42 3.55	51 6.03	93 4.59
6	22 1.86	19 2.25	41 2.02
7	6 0.51	10 1.18	16 0.79
Total	1,182 100.00	846 100.00	2,028 100.00

Crosstab with no frequencies (`nofreq`) and row-percentages (`row`):

```
tab var3 var4, nofreq row
```

var3	var4 0	1	Total
1	54.44	45.56	100.00
2	60.83	39.17	100.00
3	57.75	42.25	100.00
4	61.93	38.07	100.00
5	45.16	54.84	100.00
6	53.66	46.34	100.00
7	37.50	62.50	100.00
Total	58.28	41.72	100.00

Conditional crosstab with percentages: crosstab of var3 and var4 only when var5 only equals 0:

```
tab var3 var4 if var5==0, col
```

var3	var4 0	1	Total
1	69	51	120
	10.85	10.97	10.90
2	243	163	406
	38.21	35.05	36.88
3	220	169	389
	34.59	36.34	35.33
4	82	48	130
	12.89	10.32	11.81
5	22	34	56
	3.46	7.31	5.09
Total	636	465	1,101
	100.00	100.00	100.00

Conditional crosstab: crosstab of var3 and var4 when var5 does not equal 0

```
tab var3 var4 if var5~=0
```

var3	var4 0	1	Total
1	100	68	168
2	225	128	353
3	179	148	327
4	62	50	112
5	15	23	38
Total	581	417	998

Bivariate Statistics

Tabular Chi-Squared Tests

If you want to look to see if the cells of the crosstab are independent of one another—that there is a statistically significant relationship between the variables—a tabular chi-squared significance test is conducted by adding the option "**chi2**" at the end of the crosstab command.

```
tab var3 var4, col nofreq chi2
```

var3	var4 0	1	Total
1	16.58	19.39	17.75
2	25.89	23.29	24.80
3	21.74	22.22	21.94
4	29.86	25.65	28.11
5	3.55	6.03	4.59
6	1.86	2.25	2.02
7	0.51	1.18	0.79
Total	100.00	100.00	100.00

Pearson chi2(6) = 16.4871 Pr = 0.011

The results of the chi-squared test are presented below the crosstab. In the example above, the chi-squared statistic is 16.4871 and is statistically significant (Pr=0.011) indicating that the cells of the crosstab are, overall, significantly different from one another.

Mean Comparison t-tests

To do a mean-comparison t-test you need two variables: (1) a dependent variable and (2) a group variable. The group variable must be dichotomous (take only two values: 0 and 1) and the values should indicate which group the observation is in.

Let's examine var3 (the dependent variable):

```
sum var3
```

Variable	Obs	Mean	Std. Dev.	Min	Max
var3	2099	2.568842	1.009677	1	5

Now let's examine var4 (the group variable):

```
tab var4
```

var4	Freq.	Percent	Cum.
0	1,217	57.90	57.90
1	885	42.10	100.00
Total	2,102	100.00	

Now let's do the t-test:

```
ttest var3, by(var4)
```

Two-sample t test with equal variances

Group	Obs	Mean	Std. Err.	Std. Dev.	[95% Conf.	Interval]
0	1217	2.516845	.0278995	.9732883	2.462108	2.571581
1	882	2.64059	.0354964	1.054188	2.570922	2.710257
combined	2099	2.568842	.0220382	1.009677	2.525623	2.612061
diff		-.1237449	.0445776		-.2111657	-.036324

diff = mean(0) - mean(1) t = -2.7759
Ho: diff = 0 degrees of freedom = 2097

Ha: diff < 0 Ha: diff != 0 Ha: diff > 0
Pr(T < t) = 0.0028 Pr(|T| > |t|) = 0.0056 Pr(T > t) = 0.9972

The output presents the results of the t-test. The H0 (the null hypothesis) is that the difference in means between the groups is 0 (H0: diff=0). The alternative hypothesis that we are interested in is that the difference in the means between the groups is not 0 (Ha: diff !=0). If the t-test is significant, we can reject the null hypothesis in favor other alternative hypothesis.

t = the t-statistic

Ha: diff !=0 = The result of your t-test. If p is less than .05 than then we can conclude that the difference in means is statistically significant. In the example above, Pr(|T| > |t|) = 0.0056. The probability is less than .05 so we can conclude that treatment (var4) results in a statistically significant difference in the dependent variable (var3).

The output also presents summary statistics for the dependent variable divided into the two treatment groups:

Group = The categories of the group variable (in this case, var4)

Obs = The number of observations in each group and the groups combined

Mean = The mean of the dependent variable in each group and the groups combined

Std. Err. = The standard error of the mean of the dependent variable for each group and the groups combined

Std. Dev = The standard deviation of the dependent variable for each group and the groups combined

95% Conf. Interval = The lower and upper confidence limits of the means (assuming 95% confidence)

Correlation Statistics

The standard bivariate correlation coefficient (Pearson's r) is conducted using the `pwcorr` command. To find the correlation between two variables:

```
pwcorr var6 var7
```

	var6	var7
var6	1.0000	
var7	0.2475	1.0000

You are not restricted to only two variables. Inputting more than two variables simply produces a larger correlation matrix.

```
pwcorr var6 var7 var3
```

	var6	var7	var3
var6	1.0000		
var7	0.2475	1.0000	
var3	-0.1087	-0.0774	1.0000

Like other commands, correlations can be done conditionally. If, for example, you want to see the correlation between var6 and var7 when var3 is greater than 1, the command would be:

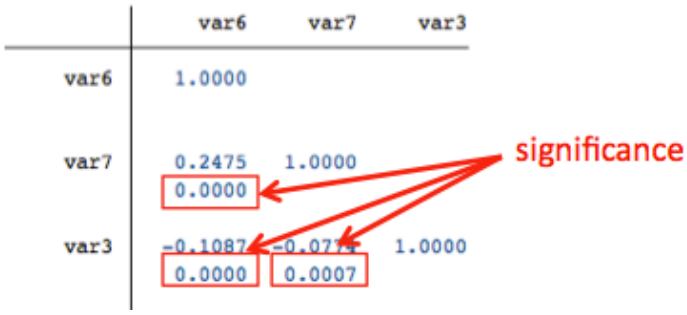
```
pwcorr var6 var7 if var3>1
```

	var6	var7
var6	1.0000	
var7	0.2752	1.0000

Correlation Significance

To determine if the correlation coefficients are statistically significant, add the `sig` option at the end. The significance level of each correlation is then reported under each correlation coefficient. In the example below, all the relationships are statistically significant.

```
pwcorr var6 var7 var3, sig
```

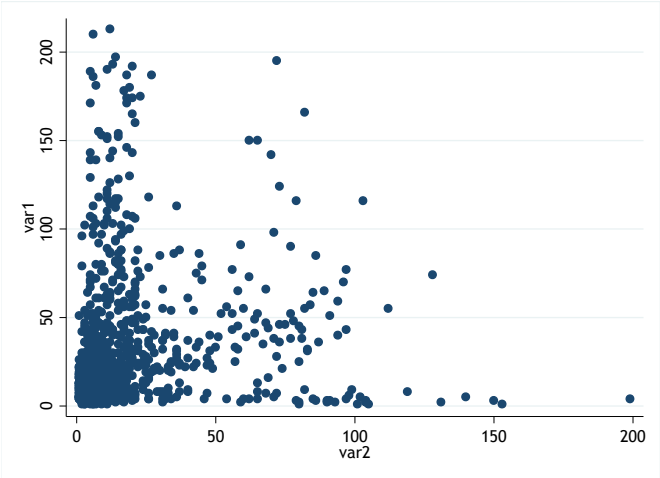


TWOWAY (X, Y) GRAPHS

Scatter Plot

The command **twoway** tells Stata that you are going to do a twoway graph. The command **scatter** tells Stata that the type of twoway you graph you want is a scatter plot. The first variable listed is the Y-coordinates and the second variable listed in the x-coordinates.

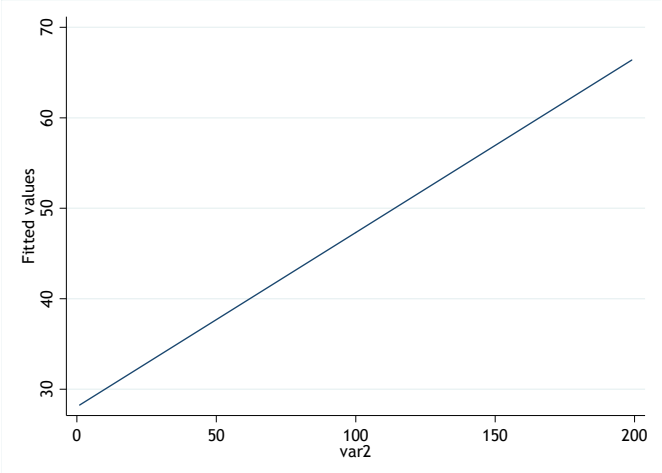
```
twoway scatter var1 var2
```



Best-Fit Line Graphs

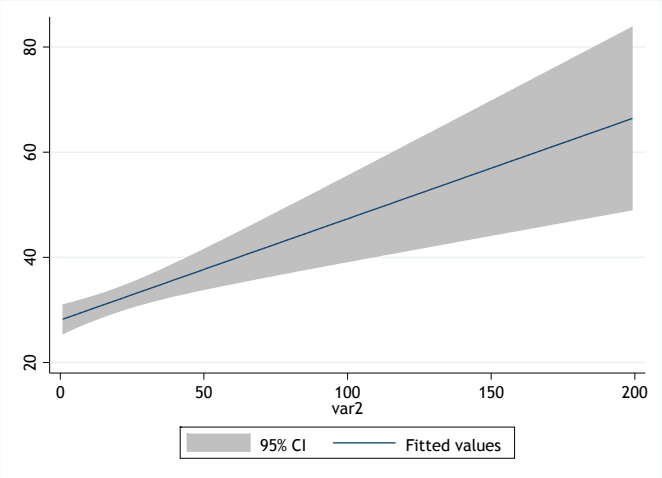
Standard best-fit line graph: The first variable listed is the dependent variable and the second variable listed in the independent variable.

```
twoway lfit var1 var2
```



Best-Fit Line with 95% confidence interval

```
twoway lfitci var1 var2
```

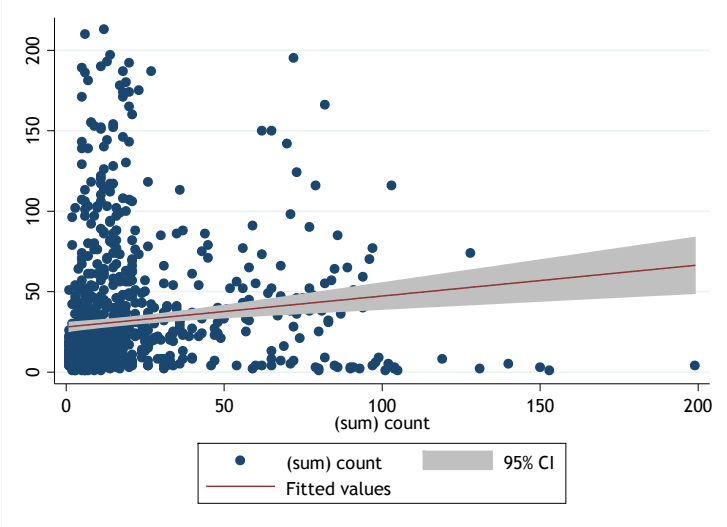


Overlaid Graphs

Stata can put multiple graphs on top of one another. Note the use of the || to separate the commands for the two graphs.

Scatter plot with best-fit line overlaid on top:

```
twoway scatter var1 var2 || lfitci var1 var2
```



OLS Regression

The basic regression command in Stata is **reg**. The command is followed first by your dependent variable and then by your independent variable(s). Options can be added on after the independent variable (following a comma).

Bivariate Regression

To regress one independent variable on a dependent variable:

Dependent Variable: var3
Independent Variables: var4 var6 var7 var8 var9 var10

```
reg var6 var7
```

analysis of variance

Source	SS	df	MS
Model	51810.9421	1	51810.9421
Residual	794142.335	1913	415.129292
Total	845953.277	1914	441.981859

Number of obs	= 1915
F(1, 1913)	= 124.81
Prob > F	= 0.0000
R-squared	= 0.0612
Adj R-squared	= 0.0608
Root MSE	= 20.375

model statistics

dv	var6	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Interval]
iv	var7	.2263023	.0202568	11.17	0.000	.1865746	.2660299
constant	_cons	49.48624	1.149554	43.05	0.000	47.23173	51.74075

model results

The upper-left of the output presents the analysis of variance for the model, the residuals, and overall:

SS = Sum of Squares
df = Degrees of Freedom
MS = Mean Square

The upper-right presents the model statistics:

Number of obs = The number of observations in the analysis (n). Remember that any observation that contains a missing value for any of the variables in the analysis (in this case, var6 and var7) will be dropped.
F = F-statistic: used to test the hypothesis that the model is significantly than the null model (no independent variables)
Prob > F = Significance of the F-statistic
R-squared = R-squared: the percent of the variance of the dependent variable accounted for by the independent variable(s)
Adj R-Squared = R-squared adjusted for the additional explanatory power that adding independent variables to a model provides.
Root MSE = Root Mean-Squared Error: another goodness-of-fit measure.

The bottom half presents the model result. You are reminded of the dependent variable (var6) in the upper right. Across the rows are the statistics for the independent variable(s) and the constant/y-intercept (_cons).

Coef. = The coefficient (β)

Std. Err. = Standard error of the coefficient

t = t-statistic for the coefficient

P>|t| = The probability that the t-statistic (and thus the coefficient) is statistically significant

[95% Conf. Interval] = 95% confidence interval for the coefficient.

Multiple Regression

Multiple regression is conducted the same way as bivariate regression; however, instead of putting one variable after the dependent variable, you put multiple variables after the dependent variable. In the example below, the independent variables var4 var6 var7 var8 var9 and var10 are regressed on the dependent variable var3.

Dependent Variable: var3
Independent Variables: var4 var6 var7 var8 var9 var10

```
reg var3 var4 var6 var7 var8 var9 var10
```

Source	SS	df	MS
Model	82.3333754	6	13.7222292
Residual	3047.92254	1853	1.64485836
Total	3130.25591	1859	1.68383858

Number of obs	1860
F(6, 1853)	8.34
Prob > F	0.0000
R-squared	0.0263
Adj R-squared	0.0231
Root MSE	1.2825

var3	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Interval]
var4	.1264638	.0630315	2.01	0.045	.0028435	.2500841
var6	-.0045343	.0015417	-2.94	0.003	-.0075579	-.0015108
var7	-.0016571	.0013703	-1.21	0.227	-.0043447	.0010304
var8	-.0057871	.0016408	-3.53	0.000	-.0090051	-.002569
var9	-.0845752	.028845	-2.93	0.003	-.1411474	-.0280031
var10	.0007805	.0164798	0.05	0.962	-.0315405	.0331015
_cons	3.934045	.1672546	23.52	0.000	3.606018	4.262073

The output is the same for multiple regression and bivariate regression. In the example above, note that 4 of the 6 independent variables are significant at the .05 level (var4, var6, var8, var9) and 2 of the 6 independent variables are statistically insignificant (var7, var10).

Regression with the if Command

You can use the `if` command when you want to run a conditional regression. Note that the `if` goes before the comma if options are being added. For example, if you wanted the same regression as above but only on the observations for whivh var10 takes a values greater than 2, the command would be:

```
reg var3 var4 var6 var7 var8 var9 var10 if var10>2
```

Source	SS	df	MS		
Model	20.5218201	6	3.42030335	Number of obs	= 211
Residual	296.302824	204	1.45246483	F(6, 204)	= 2.35
				Prob > F	= 0.0320
Total	316.824645	210	1.50868878	R-squared	= 0.0648
				Adj R-squared	= 0.0373
				Root MSE	= 1.2052

var3	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Interval]
var4	.272183	.1770381	1.54	0.126	-.076876	.6212421
var6	-.0064212	.0044206	-1.45	0.148	-.0151372	.0022948
var7	.0031183	.0041055	0.76	0.448	-.0049763	.0112129
var8	-.010769	.0052447	-2.05	0.041	-.0211098	-.0004281
var9	-.0367147	.1105293	-0.33	0.740	-.254641	.1812116
var10	.0183228	.0233335	0.79	0.433	-.0276829	.0643285
_cons	3.802284	.5617925	6.77	0.000	2.69462	4.909948

OLS with Robust Standard Errors

One common modification on the standard OLS estimator is the use of "robust standard errors." Robust standard errors are meant to overcome the problems of heteroskedasticity (see below). To estimate an OLS regression with robust standard errors, add the option `robust` at the end of the command.

```
reg var3 var4 var6 var7 var8 var9 var10, robust
```

Linear regression

Number of obs	= 1860
F(6, 1853)	= 7.67
Prob > F	= 0.0000
R-squared	= 0.0263
Root MSE	= 1.2825

var3	Coef.	Robust Std. Err.	t	P>|t|	[95% Conf.	Interval]
var4	.1264638	.0638943	1.98	0.048	.0011514	.2517762
var6	-.0045343	.0017179	-2.64	0.008	-.0079036	-.001165
var7	-.0016571	.0015397	-1.08	0.282	-.0046769	.0013626
var8	-.0057871	.0018696	-3.10	0.002	-.0094538	-.0021203
var9	-.0845752	.030304	-2.79	0.005	-.1440088	-.0251416
var10	.0007805	.0127433	0.06	0.951	-.0242122	.0257731
_cons	3.934045	.1711653	22.98	0.000	3.598348	4.269742

This option does not affect the coefficients; it only affects the standard errors and consequently whether or not the coefficient is statistically significant. An oversimplification of the procedure is that it inflates the standard errors making it more difficult to achieve statistical significance. Note that when you estimate models with robust standard errors, you don't get the analysis of variance statistics.

OLS with Clustered Standard Errors

[For more advanced users]

It is often the case that the observations in our data are not independent or come from different functional categories (people live in the same city, congressman elected in the same year, cities that are in the same state, etc.). This phenomenon can cause problems in the analysis. Stata offers many fairly advanced techniques for dealing with these problems. However, one of the relatively simple techniques is clustered standard errors in which Stata makes a correction based on a variable that identifies the groups from which observations are not independent.

For example, imagine you are doing an analysis of people after the 2008 election and their attitudes about the presidential campaigns. Because the campaign played out differently in different states, you might "cluster on states." First, identify the variable that indicates which state each observation is in (let's call that group variable "st"). Second, run your regression with the option **`cluster(st)`** at the end.

```
reg var3 var4 var6 var7 var8 var9 var10, cluster(st)
```

OLS with Fixed Effects

[For more advanced users]

Another technique for dealing with (often, more serious) non-independence problems is fixed effects. This is simply the inclusion of a dummy variable for each group (excluding one) in the regression. This can be done manually by adding the dummy variables into the model. Alternatively, you can let Stata do it for you.

```
xtreg var3 var4 var6 var7 var8 var9 var10, fe i(st)
```

Note that the **`xtreg`** command is used here instead of the **`reg`** command. **`xtreg`** is a special set of more advanced regression commands of which fixed-effects is one of them. The option **`fe`** tells Stata to use fixed effects. The option **`i(st)`** tells Stata which variable identifies the groups—in this case "st."

OLS Regression Post-Estimation

Stata provides a variety of post-estimation commands that allow you to evaluate your model and make predictions. These commands are executed after running the model. Consequently, when you run the test, Stata runs it on the most recently estimated regression.

Heteroskedasticity Test

Performs the Breush-Pagan/Cook-Weisberg test for heteroskedasticity (non-constant variance in the errors). The null-hypothesis (H0) for the test is constant variance so a significant statistic indicates that we reject the null hypothesis and there is heteroskedasticity.

```
reg var3 var4 var6 var7 var8 var9 var10
estat hettest
```

```
Breusch-Pagan / Cook-Weisberg test for heteroskedasticity
         Ho: Constant variance
         Variables: fitted values of var3

         chi2(1)      =     2.47
         Prob > chi2  =   0.1163
```

In the example above, the Prob > chi2 (0.1163) is greater than 0.05 indicating that we cannot reject the null hypothesis so there is constant variance (no heteroskedasticity or homoskedasticity in the errors).

Omitted Variable Test

Performs the Ramsey RESET test for omitted variables. The null-hypothesis (H0) for the test is no omitted variables so when the statistic is significant it indicates that there are omitted variables.

```
reg var3 var4 var6 var7 var8 var9 var10
estat hettest
```

```
Ramsey RESET test using powers of the fitted values of var3
       Ho:  model has no omitted variables
                F(3, 1850) =      1.49
                  Prob > F =      0.2162
```

In the example above, the Prob > chi2 (0.2162) is greater than 0.05 indicating that we cannot reject the null hypothesis and that there are no omitted variables. Note that this test should be treated with a certain amount of caution as most statistical models produced in the social sciences suffer from some amount of omitted variable bias.

Linear Predictions

Stata can calculate the linear prediction for each observation (y-hat). This is what you get when you use the actual data for each observation and the estimated coefficients to compute Y.

To calculate the prediction, first run the model.

```
reg var3 var4 var6 var7 var8 var9 var10
```

Then ask for the predictions. The command **predict** tells Stata that you want it to create a new variable that contains a prediction for each observation based on the most recent model. You then name the variable (in this case, yhat_m1, but you can call it whatever you want). Finally, you tell it what kind of prediction you want. The option **xb** tells it to produce linear predictions.

```
predict yhat_m1, xb
```

Residual Predictions

Stata can calculate the residual for each observation based on the previous model: the distance between the actual value of Y and the predicted value of Y (y-hat). The procedure is similar to the linear prediction command. After you run the model, you do the following command.

```
predict e, resid
```

In this example, Stata will produce a new variable named "e" that contains the residuals for each observation for the most recent model.

Marginals

[For more advanced users]

Stata has a nice function that allows you to estimate marginals: predictions with standard errors based on different values of an independent variable of interest while holding the other independent variables constant. The function is more complex than many of the others covered in this guide and best understood by way of example. Imagine you want to model the following variables:

Dependent Variable:
var3 = ordinal variable taking values 1, 2, 3, 4, 5, 6, and 7 with mean 2.8

Independent Variables:
var4 = dummy variable with mode 0
var6 = continuous variable ranging from 0 to 100 with mean 61.3
var8 = continuous variable raning from 0 to 100 with mean 74.5
var9 = ordinal variable taking values 1, 2, 3, 4, and 5 with mean 4.2 and median 5

First, run a regression:

```
reg var3 var4 var6 var8 var9
```

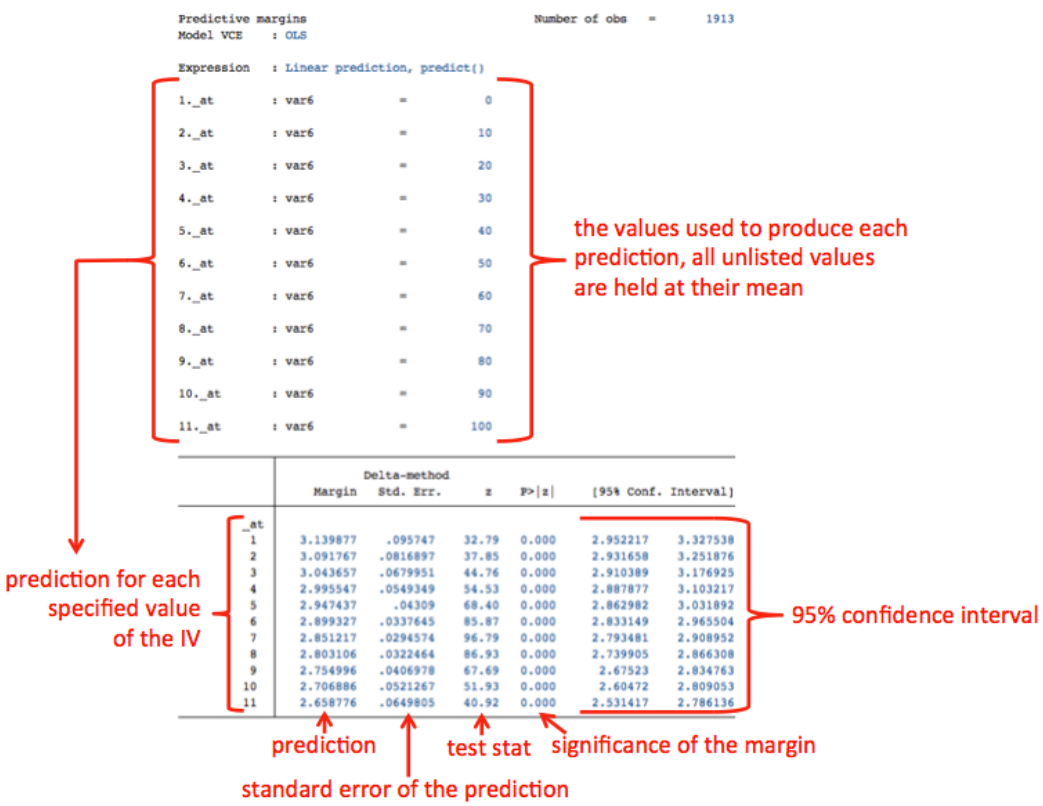
Source	SS	df	MS
Model	78.5705389	4	19.6426347
Residual	3156.93809	1908	1.65457971
Total	3235.50863	1912	1.69221162

Number of obs	1913
F(4, 1908)	11.87
Prob > F	0.0000
R-squared	0.0243
Adj R-squared	0.0222
Root MSE	1.2863

var3	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Interval]
var4	.1021005	.0619772	1.65	0.100	-.0194496	.2236507
var6	-.004811	.0014906	-3.23	0.001	-.0077344	-.0018876
var8	-.0059636	.0016038	-3.72	0.000	-.009109	-.0028183
var9	-.0818517	.0284935	-2.87	0.004	-.1377335	-.02597
_cons	3.884538	.1623623	23.93	0.000	3.566111	4.202964

Seeing that var6 is significant, it might be helpful to better understand its relationship with the dependent variable var3 when the other 3 independent variables are held constant. We will look at var6 at 0, 10, 20, 30, 40, 50, 60, 70, 80, 90, and 100. When we hold the other three variables constant, we either hold them at their mean, their median, their mode, or some value of interest. Let's start by holding all the other independent variables at their mean. To do this:

```
margins, at(var6=(0(10)100) (mean) _all)
```



The part of the command `var6=(0(10)100)` asks Stata to produce margins for var6 for values of var6 beginning at 0 and increasing by 10 until 100. If we wanted values between 1 and 2 increasing by tenths, the command would be margins, at(var6(1(.1)2)). The part of the command `(mean) _all` tells Stata to hold all other variables at their mean (note that this command can be excluded as Stata's default is to hold all other variables at their mean).

The Margins column reports Y-hat given different values of your IV of interest:

When var6 equals 0 and all other variables are held at their mean, Y-hat=3.13. The confidence interval tells us that we can be 95% confident that the Y-hat falls between 2.95 and 3.32.

When var6 equal 10 and all other variables are held at their mean, Y-hat=3.09. The confidence interval tells us that we can be 95% confident that the Y-hat falls between 2.93 and 3.25.

The same thing can be done for each of the values of var6 that we asked for.

The command does not require you to only hold the other independent variables at their means. Sometimes you will want to hold them at different values. In fact, some argue that we should only hold variables at plausible values. For example, var4 is a dummy variable and as such can only take

the values 0 or 1 so you may want to hold it constant at 0 or 1. Others argue that by holding a dummy variable at .5 or its mean, you effectively neutralize the effect of the variable (which is the purpose of holding it constant).

To get the margins for var6 while holding var4 at its mode, var8 at its mean, and var9 at its median, the command is:

```
margins, at(var6=(0(10)100) var4=0 (mean) var8 (median) var9)
```

Note that Stata does not allow you to simply tell it to do the mode for var4 the way it does for means and medians. Instead, we simply set var4 to the mode with var4=0. If you wanted to do the same command as above but set both var8 and var9 to their median the command would be:

```
margins, at(var6=(0(10)100) var4=0 (median) var8 var9)
```

Advanced Regression Techniques

This section very briefly covers a set of more advanced regression techniques called generalized linear models. For more information on how to use the commands and how to interpret the outputs, I recommend:

Hoffmann, J. P. 2003. *Generalized Linear Models: An Applied Approach*. Allyn & Bacon.

Regression with Dichotomous Dependent Variables: Logits and Probits

The two standard regression techniques for dichotomous dependent variables are the Logit and Probit. Running Logit and Probit models in Stata is strait-forward.

Logit:

```
logit var4 var6 var3
```

Probit:

```
probit var4 var6 var3
```

Regression with Ordinal Dependent Variables: Ordered Logit

While OLS can be used for ordinal dependent variables, ordered logit is considered to be a better technique as it does not allow for estimates that are between categories.

```
ologit var9 var6 var4
```

Regression with Unordered Categorical Dependent Variables

The standard regression technique for unordered categorical dependent variables is the multinomial logit. This model also requires you to specify an excluded category for the dependent variable.

```
mlogit var10 var4 var6 var9, baseout(2)
```

The option **baseout** tells Stata which category of the dependent variable to exclude for comparison purposes. In the example above, Stata will exclude the category of the dependent variable (var10) that takes the value 2.