Categorical Data Analysis

Getting Started Using Stata

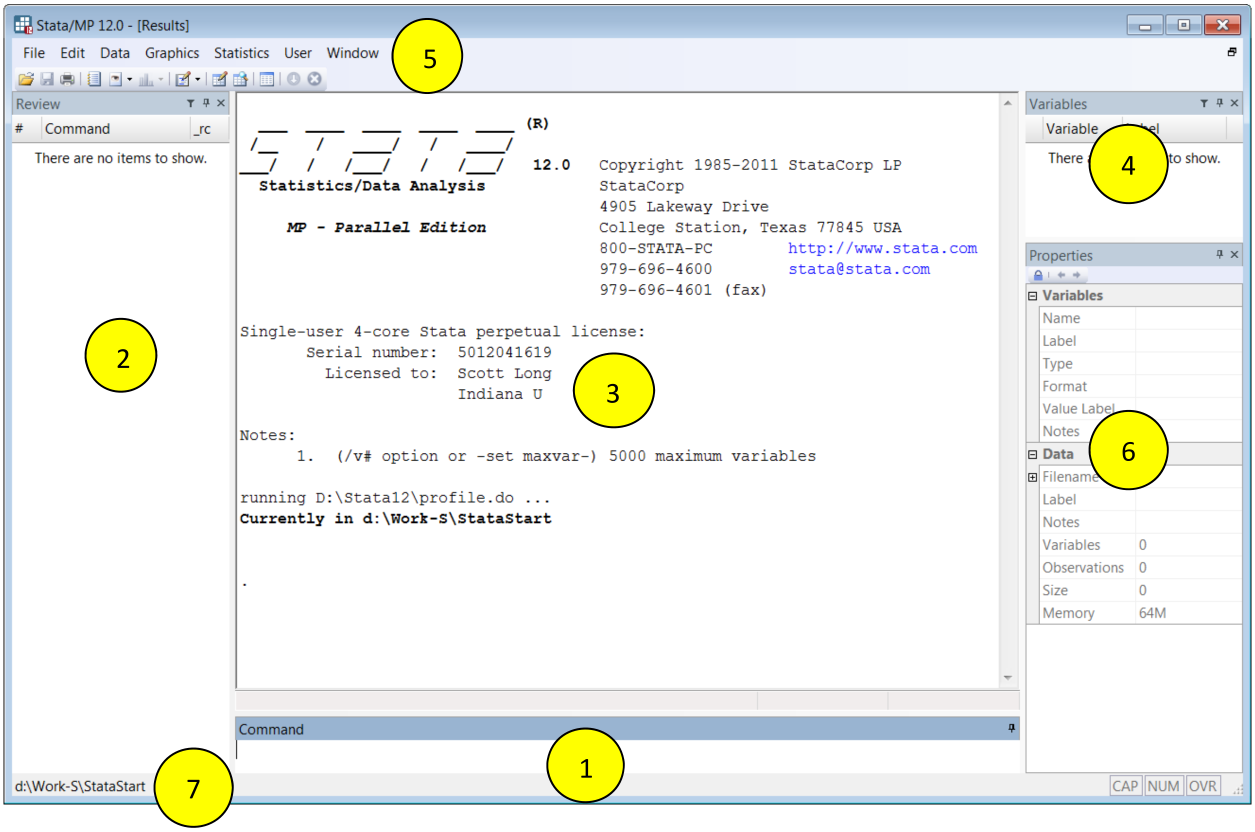
Scott Long and Shawna Rohrman

cda12 StataGettingStarted 2012-05-11.docx

Getting Started in Stata

Opening Stata

When you open Stata, the screen has seven key parts (This is Stata 12. Some of the later screen shots are from earlier versions of Stata):



```
  ___  ____  ____  ____  ____ (R)
 /__    /   ____/   /   ____/
___/   /   /___/   /   /___/   12.0   Copyright 1985-2011 StataCorp LP
  Statistics/Data Analysis            StataCorp
                                      4905 Lakeway Drive
     MP - Parallel Edition            College Station, Texas 77845 USA
                                      800-STATA-PC        http://www.stata.com
                                      979-696-4600        stata@stata.com
                                      979-696-4601 (fax)

Single-user 4-core Stata perpetual license:
       Serial number:  5012041619
         Licensed to:  Scott Long
                       Indiana U

Notes:
      1.  (/v# option or -set maxvar-) 5000 maximum variables

running D:\Stata12\profile.do ...
Currently in d:\Work-S\StataStart


.
```

1. The Command Window

This is one place where you can enter commands. Try typing `sysdir` into the Command Window, and then press *enter*. In the area above the Command Window, you'll see Stata has recognized the command and given you a response. More on that later. There are some shortcut keys associated with the Command Window: PAGE UP, PAGE DOWN, and the TAB key. PAGE UP and PAGE DOWN will allow you to scroll through the commands you've already entered into the Command Window. Try PAGE UP: the `sysdir` command should come up again. When the Command Window is blank, think of yourself at the bottom of the list; the PAGE UP key will allow you to navigate up the list, and then you use the PAGE DOWN key to get back down the list. The TAB key completes variable names for you. If you enter the first few letters of a variable name and then press TAB, Stata will fill in the rest of the variable name for you, if it can.

2. The Review Window

When you enter a command in the Command Window, it appears in the Review Window. If you look now at the Review Window, it should say “1   sysdir”. Stata numbers the list of commands you execute and stores them in the Review Window. If you wish, you can clear this window by right-clicking on it and selecting clear. (This window can be very helpful for you, so consider whether you might need those commands later before you clear them out.) Clicking once on a command enters it into the Command Window. Double-clicking a command tells Stata to execute this command. Additionally, you can send commands stored in the Review Window to your do-file (a file you’ll use to do programming for this class—instead of using the point-and-click features of Stata, we will write our commands into Stata’s do-file). This means that if you’re experimenting with a particular command, you can play around in the Command Window first, and then once you’ve gotten the options you want you can send it right to the do-file. Let’s try it: type `doedit` in the Command Window to open a new do-file, then right click the `sysdir` command and send it to the do-file.

3. The Results Window

The Results Window is where all of the output is displayed. When you execute a command—whether through the Command Window, do-file editor, or the Graphical User Interface (GUI)—the results appear here. As you saw when we typed in `sysdir`, Stata retrieved a list of the program’s system directories. If your command takes up the whole Results Window, Stata will need to be prompted to continue. You’ll see a blue “—more—,” indicating there is more output to view. To see more, either click on “—more—,” or you can enter a space into the Command Window. You can scroll up in the Results Window to see previous output, but if you’ve been working for a while, the scroll buffer may not be large enough to go all the way back to the beginning. You can fix this: Edit→ Preferences → General Preferences → Windowing. The default buffer size is 32,000 bytes, but increasing this to 500,000 bytes should allow you to go back to most of your output. (Note: You may have to restart Stata for this to go into effect.)

4. The Variable Window

Once you’ve loaded data, the Variable Window will show you the variable’s name and label, the variable type, and the format of the variable. If using the Command Window, you can click on variable names to enter them in the Command Window (it doesn’t matter if you single- or double-click, both will display the variable’s name in the Command Window). Later in this guide, you’ll learn how to rename, label, and attach notes to your variables in the do-file. However, the option to do these tasks is also available by right-clicking on the variable name.

5. The Toolbar

Open a dataset.

Save the dataset you’re working on.

Print any of the files you have open: the dataset you’re working on, do-file you have open, etc.

Begin/Close/Suspend/Resume a Log (see next section)

Open the Viewer (you'll use this mainly to get help).

Bring a graph to the front (you'll be able to choose from whatever graphs you have open).

Open the do-file editor

Open the data editor. Here, you can edit the dataset.

Browse the dataset. No editing capabilities.

Prompts Stata to continue displaying output when the command fills the window. This has the same effect as entering a space into the Command Window.

Stops the current command(s) from being estimated.

6. The Properties Window

Once you've loaded data, the Properties Window will show you information on data and variables. For variables, you will see a highlighted variable's name, label, and notes plus its type and format. For data, you will see the filename and the path to the data plus labels, notes, the number of variables, and several more useful bits of information about the data. If you click on the lock at the top of this window, you can edit information. For example, you can change a variable's name. This will also allow you to add notes and labels to both variables and data. If you click on the arrows at the top, you can also scroll the variables in the dataset.

7. The Working Directory

In the bottom left corner of Stata, you will notice a path to a folder on your computer. In the screenshot above this path is "d:\Work-S\Stata-Start." Your working directory will be different. This location is where you will keep all your do files, data, graphs, log files, etc. related to a project. This path directs Stata where to looks for information like data and save information like log files. It is very important that you set your working directory every time you open Stata. You can set your directory by using the command window to type: `cd "c:\PATH-TO-FOLDER"` and then hit your enter key. For more information, see the section "Setting Your Working Directory."

Do Files and Log Files

As mentioned above, Stata can be used through the Graphical User Interface or by entering commands in do-files. In this class, we will be using do-files. Do-files are basically text files where you can write out and save a series of Stata commands. When you set up the do-file, you'll also set up a log file, which stores Stata's output. To open the do-file editor, type `doedit` into the Command Window. Here is an example of how to set up your do-file:

```
1>  capture log close
```

```
 2>  log using cda12-stataintro-template, replace text
 3>
 4> //  program:    cda12-stataintro-template.do
 5> //  task:       Getting started using Stata: template for do-file
 6> //  project:    CDA
 7> //  author:     Scott Long 2011-08-25
 8>
 9> //  #1
10> //  program setup
11>
12> version 12
13> clear all
14> matrix drop _all
15> set linesize 80
16>
17> //  #2
18> //  load data
19>
20> //  #3
21> //
22>
23> log close
24> exit
```

Line 1 closes any log files that might already be open, so Stata can start a new log file for the current do-file. Line 2 opens a new log-file with the same name as the do-file. This way, there should always be a pair of do-files and log-files with the same name. We tell Stata to replace this file if it already exists (this allows you to update the file if you need to make changes), and asks that the format of the file be a text file. The default format for the Stata log-file is SMCL, but the text files are more versatile.

Lines 4-7 are important for internally documenting your do-file. They indicate the name of the do-file, the tasks for this do-file, the overall project you're working on, and your name and date. This heading is especially helpful if you print results because you will know where the output came from, the project it's for, and the date. Line 12 specifies the version of Stata used to run the do-file. If you run this do-file on a later version of Stata, say Stata 13, specifying `version 12` allows you to get the same results you obtained using Stata 12. Lines 13 and 14 clear out existing data and matrices so there is nothing left in Stata's memory. This allows the current do-file to run on a clean slate, so to speak. The number of characters in each line of Stata's output is set by line 15. You start your commands at line 17, where you'll need to load the data. Insert as many lines needed to complete your do-file. At the end of the file, be sure to include the commands `log close` (line 23) and `exit` (line 24). These commands close the log file, and tell Stata to terminate the do-file. With the `exit` command, Stata will not read the do-file any further. This is sometimes a handy place to keep notes or to-do lists.

Notice that some lines begin with two forward slashes. This tells Stata that anything that follows are comments, not commands to execute. These are important for documenting the do-file. You can also "comment out" lines in your do-file by placing an asterisk (*) at the beginning of each line. Additionally, if you want to include extensive comments, you can use `/*` to begin the comments and `*/` to close them. Finally, your commands may be more than 80 characters long—for instance, when you use graphs later in the course. When this happens, you will need to use three forward slashes at the end of each line to signify that the command carries onto the next line.

Note: If you would like more detailed information about organizing do-files, see *The Workflow of Data Analysis Using Stata*.

Setting your Working Directory

Note*: Your working directory should already be set here in the lab. However, the instructions given here will tell you how to set a working directory from your personal or office computer/outside this lab.*

When using datasets in Stata, you'll most often open the dataset from a file on your computer (i.e., with the `use` command). In order to do that, you must enter the pathname of the data file into the do-file. If you switch computers—as you might in this class—the data's pathname might be different on one computer than it is on another. For instance, if you use an external hard drive or a flash drive, it might be drive E on one computer and F on another. To fix it, you'll have to change the pathname in the do-file each time you want to use that dataset. To avoid having to do this, you can set the folder you're using as a **working directory** at the beginning of each Stata session. Then, all you'll need to do is refer to the dataset by its filename without any path. The other benefit of the working directory is that when you use do-files and log files, Stata will save the log files in your working directory. (It does not matter where your do-file is saved, but for the sake of organization, it helps if the do-file is in the same folder as the data.) You'll know where Stata saved the file, because it will show the current working directory in the lower left corner of the window. See bubble #7 in the screen shot of Stata above; it shows that the working directory is D:\Work-S\StataStart\.

You can also check the path to the current working directory this way:

```
. pwd
c:\stata_start
```

To change your working directory usedi`cd`. If there are spaces in the pathname, you'll need to put double quotes around the pathname.

```
. cd "E:\My Documents\Classes\CDA"
E:\My Documents\Classes\CDA
```

Now, when I want to use a dataset, all I need to do is enter `use` ***`dataset-name`*** and Stata will look for it in my working directory:

```
. use cleaned-scireview4
(Biochemist data for review - Some data artificially constructed)
```

The *Workflow* book has more detailed information on this, as well as more advanced ways to set up working directories.

Installing User-written Packages

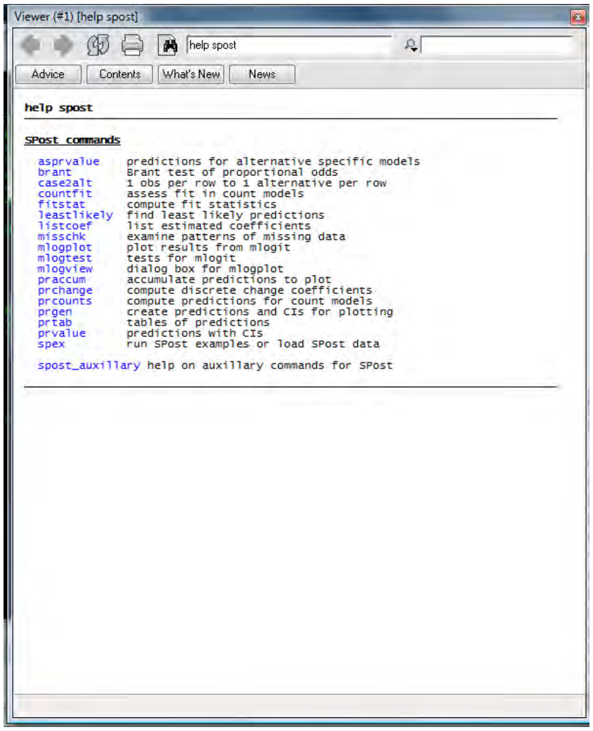
In addition to Stata's base packages, there are many auxiliary Stata packages available to download. The packages used in this course include SPost. While SPost might already be installed on your computer (details are given in lab), you can install these programs yourself. (Note: In public labs you will need to have Write Permissions to save files. See your local computer expert.) To install the SPost package as of August 25, 2011, enter Stata while connected to the internet. Type `findit spost9_ado` in the Command Window. A Viewer window appears that lists links for installation of the package. Read the descriptions carefully, as sometimes packages with similar names will also be included in the list. Once you select the package, the Viewer will show you a list of the files included in the package. The "Click here to install" link will install the files in the Stata directory. After downloading, try the help file for that package to make sure it was correctly installed.

Getting Help

There are help files for all of the commands and packages you'll be using in this course. To access them, you simply type help [command/package] into the Command Window. For example,

```
. help spost
```

Brings up this Viewer window:



```
help spost

SPost commands

  asprvalue    predictions for alternative specific models
  brant        Brant test of proportional odds
  case2alt     1 obs per row to 1 alternative per row
  countfit     assess fit in count models
  fitstat      compute fit statistics
  leastlikely  find least likely predictions
  listcoef     list estimated coefficients
  misschk      examine patterns of missing data
  mlogplot     plot results from mlogit
  mlogtest     tests for mlogit
  mlogview     dialog box for mlogplot
  praccum      accumulate predictions to plot
  prchange     compute discrete change coefficients
  prcounts     compute predictions for count models
  prgen        create predictions and CIs for plotting
  prtab        tables of predictions
  prvalue      predictions with CIs
  spex         run SPost examples or load SPost data

  spost_auxillary help on auxillary commands for SPost
```

Within this window, you can click links to take you to related help pages. Also, most commands have options you can use to customize output. These options, along with examples of how to use commands, will be included in the help files. Simply typing `help` will bring up this window as well, showing the contents of the help file.

Exploring your Data

Note: You can follow along with this and the next section of this guide with the do-file `cda12-stataintro.do`.

Importing/Using Data

The first thing you will need to do to begin analyzing data is to load a dataset into Stata. There are several ways to do this. The most common way is to use the `use` command to call up data saved on your computer. However, the datasets used in this class are available via Prof. Long's SPost website (http://www.indiana.edu/~jslsoc/spost.htm). In order to access them, you can use the `spex` command:

```
. spex cleaned-scireview4, clear
```

If the dataset is already in your working directory, you can:

```
. use cleaned-scireview4, clear
```

Once you load the data you can begin to explore. We start by saving the data so that we don't accidentally change the original data (you'll change the last three letters to your own initials):

```
. save cleaned-scireview4-jsl, replace
(note: file cleaned-scireview4-jsl.dta not found)
file cleaned-scireview4-jsl.dta saved
```

The `replace` option tells Stata that if this file already exists in your working directory, you want to replace it. In the output, you can see that this file did not already exist, so there was no replacement, only the creation of a new file. Now, we can clear out Stata's memory and recall the data with the `use` command.

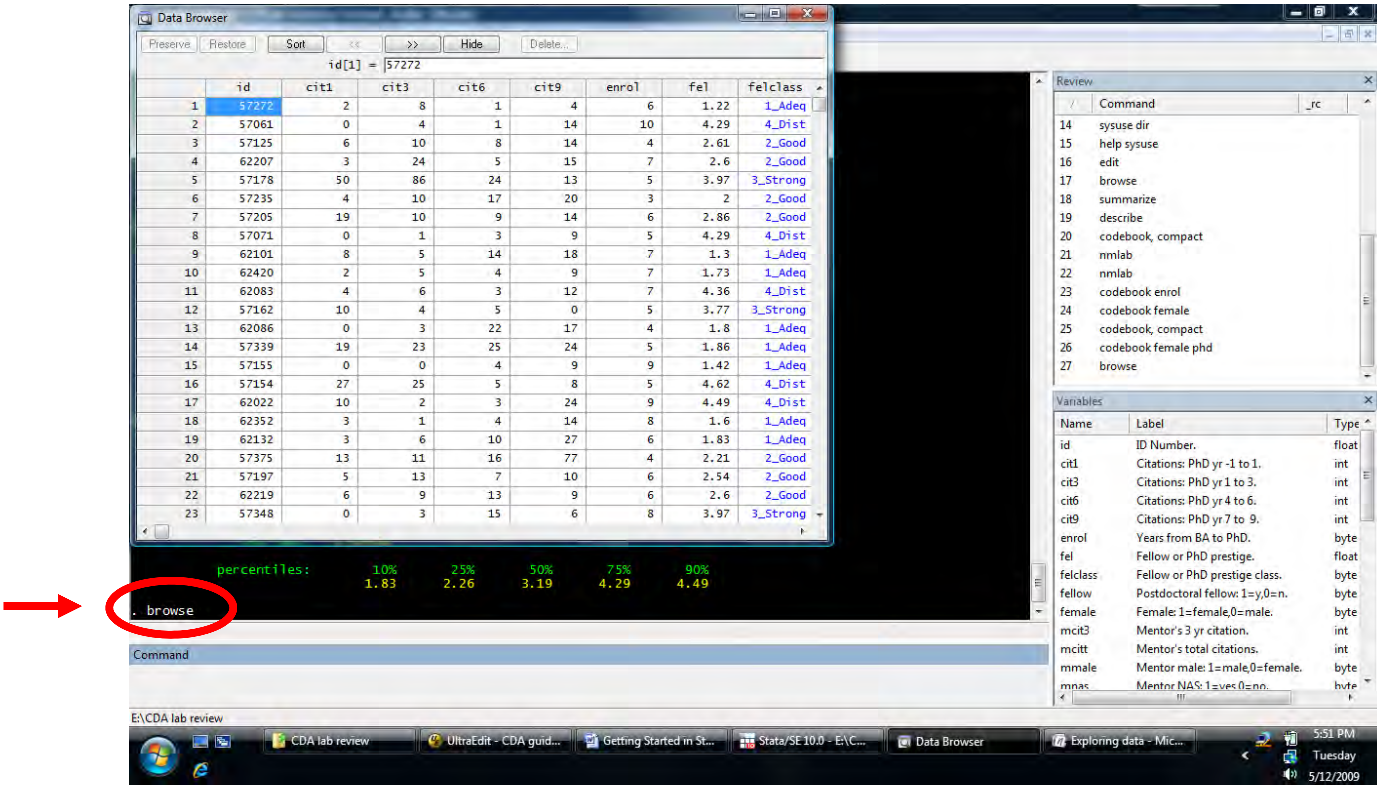
```
. use cleaned-scireview4-jsl, clear
(Biochemist data for review - Some data artificially constructed)
```

While we've provided you with the data you'll need for the course, Stata also comes with example datasets you can use. To see a list of the example datasets, type `sysuse dir.` If you want to use one of these datasets, the command is `sysuse` *`dataset-name`*. The `sysuse` help file provides more information.

When working from home, you may want to use data that is not in Stata format. Consult the *Workflow* book for more information on importing different types of data files.

Exploring Your Data

There are a variety of commands for exploring your data. First, you can look at the data in the spreadsheet format. This may be especially helpful for new Stata users who are more fluent in SPSS. To "look" at the data, use the `browse` command. You cannot edit the data using the `browse` command, so it is safer than using the `edit` command which brings up the data in spreadsheet format, but allows you to edit it as well. The following is from Stata 10:



id[1] = 57272

	id	cit1	cit3	cit6	cit9	enrol	fel	felclass
1	57272	2	8	1	4	6	1.22	1_Adeq
2	57061	0	4	1	14	10	4.29	4_Dist
3	57125	6	10	8	14	4	2.61	2_Good
4	62207	3	24	5	15	7	2.6	2_Good
5	57178	50	86	24	13	5	3.97	3_Strong
6	57235	4	10	17	20	3	2	2_Good
7	57205	19	10	9	14	6	2.86	2_Good
8	57071	0	1	3	9	5	4.29	4_Dist
9	62101	8	5	14	18	7	1.3	1_Adeq
10	62420	2	5	4	9	7	1.73	1_Adeq
11	62083	4	6	3	12	7	4.36	4_Dist
12	57162	10	4	5	0	5	3.77	3_Strong
13	62086	0	3	22	17	4	1.8	1_Adeq
14	57339	19	23	25	24	5	1.86	1_Adeq
15	57155	0	0	4	9	9	1.42	1_Adeq
16	57154	27	25	5	8	5	4.62	4_Dist
17	62022	10	2	3	24	9	4.49	4_Dist
18	62352	3	1	4	14	8	1.6	1_Adeq
19	62132	3	6	10	27	6	1.83	1_Adeq
20	57375	13	11	16	77	4	2.21	2_Good
21	57197	5	13	7	10	6	2.54	2_Good
22	62219	6	9	13	9	6	2.6	2_Good
23	57348	0	3	15	6	8	3.97	3_Strong

```
      percentiles:        10%       25%       50%       75%       90%
                         1.83      2.26      3.19      4.29      4.49

. browse
```

/	Command	_rc
14	sysuse dir	
15	help sysuse	
16	edit	
17	browse	
18	summarize	
19	describe	
20	codebook, compact	
21	nmlab	
22	nmlab	
23	codebook enrol	
24	codebook female	
25	codebook, compact	
26	codebook female phd	
27	browse	

Name	Label	Type
id	ID Number.	float
cit1	Citations: PhD yr -1 to 1.	int
cit3	Citations: PhD yr 1 to 3.	int
cit6	Citations: PhD yr 4 to 6.	int
cit9	Citations: PhD yr 7 to 9.	int
enrol	Years from BA to PhD.	byte
fel	Fellow or PhD prestige.	float
felclass	Fellow or PhD prestige class.	byte
fellow	Postdoctoral fellow: 1=y,0=n.	byte
female	Female: 1=female,0=male.	byte
mcit3	Mentor's 3 yr citation.	int
mcitt	Mentor's total citations.	int
mmale	Mentor male: 1=male,0=female.	byte
mnas	Mentor NAS: 1=yes,0=no.	byte

Names, Labels, and Summary Statistics

You want to know what variables are in the dataset. Here are two commands that will list variable names and their labels. First, the `nmlab` command:

```
. nmlab

id          ID Number.
cit1        Citations: PhD yr -1 to 1.
cit3        Citations: PhD yr 1 to 3.
cit6        Citations: PhD yr 4 to 6.
<snip>      <= this means output was deleted
jobimp      Prestige of 1st univ job/Imputed.
jobprst     Rankings of University Job.
```

This simple command gives you the name and the label of the variable. You can also use options to have Stata return variable labels to you as well (see `help nmlab`). Note that this command is part of the `workflow` package and also in `spost9_ado`.

The `describe` command is a little more detailed:

```
. describe

Contains data from cleaned-scireview4-jsl.dta
  obs:           264                          Biochemist data for review -
```

```
                                                Some data artificially
                                                constructed
 vars:            34                          12 May 2009 17:08
 size:        15,576 (99.9% of memory free)   (_dta has notes)
-------------------------------------------------------------------------------
              storage  display     value
variable name   type   format      label      variable label
-------------------------------------------------------------------------------
id              float  %9.0g                  ID Number.
cit1            int    %9.0g                  Citations: PhD yr -1 to 1.
<snip>
jobprst         float  %9.0g       prstlb     Rankings of University Job.
                                            * indicated variables have notes
-------------------------------------------------------------------------------
Sorted by:  jobprst
```

Like `nmlab`, `describe` gives you variable names and labels, but also gives information about the dataset. If you want just the information about the dataset, you would use the `short` option.

Often, you'll want to see summary statistics for your variables (e.g., means, minimum and maximum values). Both the `summarize` and `codebook, compact` commands are useful for this:

```
. summarize

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
          id |       264    58556.74        2239      57001      62420
        cit1 |       264    11.33333    17.50987          0        130
        cit3 |       264    14.68561    21.26377          0        196
<snip>
      jobimp |       264    2.864109    .7117444       1.01       4.69
     jobprst |       264    2.348485    .7449179          1          4

. codebook, compact

Variable   Obs Unique      Mean    Min    Max  Label
-------------------------------------------------------------------------------
id         264    264  58556.74  57001  62420  ID Number.
cit1       264     48  11.33333      0    130  Citations: PhD yr -1 to 1.
cit3       264     54  14.68561      0    196  Citations: PhD yr 1 to 3.
<snip>
jobimp     264    180  2.864109   1.01   4.69  Prestige of 1st univ job/Imputed.
jobprst    264      4  2.348485      1      4  Rankings of University Job.
-------------------------------------------------------------------------------
```

The two commands provide the same information, with the exception of standard deviations and variable labels. The `codebook` command, without the `compact` option, gives more detailed information about the variables in the data, including information on percentiles for continuous variables. Here is the codebook information for two variables (one binary and one continuous):

```
. codebook female phd

-------------------------------------------------------------------------------
female                                           Female: 1=female,0=male.
-------------------------------------------------------------------------------

                  type:  numeric (byte)
                 label:  femlbl
                 range:  [0,1]                        units:  1
         unique values:  2                        missing .:  0/264

            tabulation:  Freq.   Numeric  Label
                           173         0  0_Male
                            91         1  1_Female

-------------------------------------------------------------------------------
phd                                              Prestige of Ph.D. department.
-------------------------------------------------------------------------------

                  type:  numeric (float)

                 range:  [1,4.66]                     units:  .01
         unique values:  79                       missing .:  0/264

                  mean:   3.18189
              std. dev:   1.00518

           percentiles:        10%       25%       50%       75%       90%
                              1.83      2.26      3.19      4.29      4.49
```

Similarly, using the `detail` option for the `summarize` command gives more information about selected variables:

```
. summarize female phd, detail

                  Female: 1=female,0=male.
-------------------------------------------------------------
      Percentiles      Smallest
 1%            0              0
 5%            0              0
10%            0              0       Obs                 264
25%            0              0       Sum of Wgt.         264

50%            0                      Mean            .344697
                        Largest       Std. Dev.      .4761721
75%            1              1
90%            1              1       Variance       .2267398
95%            1              1       Skewness       .6535369
99%            1              1       Kurtosis        1.42711

                Prestige of Ph.D. department.
-------------------------------------------------------------
      Percentiles      Smallest
 1%            1              1
 5%         1.68              1
10%         1.83              1       Obs                 264
25%         2.26           1.22       Sum of Wgt.         264

50%         3.19                      Mean           3.181894
                        Largest       Std. Dev.       1.00518
75%         4.29           4.62
90%         4.49           4.66       Variance       1.010387
95%         4.54           4.66       Skewness       -.144854
99%         4.66           4.66       Kurtosis       1.771461
```

Listing Observations

Listing observations is another way to explore the data. Say you are interested in the characteristics of the observations with very high and very low publication records. You could list these observations. First, you'd want to sort the observations according to their total publications (Stata will automatically sort in ascending order):

```
. sort pubtot
```

Listing the five with the lowest publication record, along with their gender, PhD prestige class, their job's prestige, and the number of years enrolled in the PhD program:

```
. list id pubtot female phdclass jobprst enrol in 1/5

     +------------------------------------------------------+
     |    id   pubtot   female   phdclass   jobprst   enrol |
     |------------------------------------------------------|
  1. | 57050        0    1_Yes     2_Good    2_Good       7 |
  2. | 57031        0     0_No     2_Good    2_Good       6 |
  3. | 62151        0    1_Yes     4_Dist    2_Good       4 |
  4. | 57238        0    1_Yes     2_Good    2_Good       5 |
  5. | 57087        0     0_No     1_Adeq    2_Good       4 |
     +------------------------------------------------------+
```

The `in 1/5` statement tells Stata that you are requesting a list of observations 1 through 5. It appears that there may be more than five observations with no publications; if so, Stata will list them randomly. (This means that you may not see the observations in the same order every time.) You can specify that you want to see all individuals with no publications with an `if` statement:

```
. list id pubtot female phdclass jobprst enrol if pubtot==0

     +-------------------------------------------------------+
     |    id   pubtot   female   phdclass    jobprst   enrol |
     |-------------------------------------------------------|
  1. | 57050        0    1_Yes     2_Good     2_Good       7 |
  2. | 57031        0     0_No     2_Good     2_Good       6 |
  3. | 62151        0    1_Yes     4_Dist     2_Good       4 |
  4. | 57238        0    1_Yes     2_Good     2_Good       5 |
  5. | 57087        0     0_No     1_Adeq     2_Good       4 |
     |-------------------------------------------------------|
  6. | 62350        0     0_No     1_Adeq     2_Good       6 |
  7. | 57132        0    1_Yes     4_Dist   3_Strong       5 |
  8. | 57267        0    1_Yes     2_Good     2_Good       7 |
  9. | 62266        0     0_No     2_Good     2_Good       9 |
 10. | 57226        0     0_No     2_Good     2_Good       5 |
     |-------------------------------------------------------|
 11. | 57042        0    1_Yes     2_Good     2_Good       6 |
 12. | 57246        0    1_Yes     2_Good     2_Good       8 |
 13. | 57311        0    1_Yes     2_Good     2_Good       8 |
 14. | 57305        0     0_No     2_Good   3_Strong       5 |
     +-------------------------------------------------------+
```

When using the `if` statement, you are saying you only want Stata to return a list if a certain condition is met—in this case, if the observation's value on `pubtot` is equal to zero. Notice that the `if` statement uses a double equal sign; this double equal sign is used for equality testing. To see the top five publishers:

```
. list id pubtot female phdclass jobprst enrol in -5/L

     +-------------------------------------------------------+
     |    id   pubtot   female   phdclass    jobprst   enrol |
     |-------------------------------------------------------|
260. | 57184       46     0_No     4_Dist     4_Dist       5 |
261. | 57298       55     0_No   3_Strong   3_Strong       4 |
262. | 57043       59    1_Yes     4_Dist   3_Strong       5 |
263. | 57084       64     0_No     2_Good   3_Strong       5 |
264. | 57229       73     0_No   3_Strong   3_Strong       5 |
     +-------------------------------------------------------+
```

Here, the `in -5/L` requests the fifth-to-last observation (`-5`) through the last observation (`L`). To suppress the value labels (e.g., 4_Dist) to only show the numeric values, add the `nolabel` option to the command.

Variable Distributions

Here are some quick ways to look at the distribution of your variables. For categorical variables, use the `tabulate` command. This command will allow you to tabulate one variable on its own, or cross-tabulate it with another:

```
. tabulate female, miss

    Female? |
    (1=yes) |      Freq.     Percent        Cum.
------------+-----------------------------------
       0_No |        173       65.53       65.53
      1_Yes |         91       34.47      100.00
------------+-----------------------------------
      Total |        264      100.00
```

```
. tabulate phdclass female, miss

  Prestige |
  class of |
     Ph.D. |    Female? (1=yes)
     dept. |      0_No      1_Yes |     Total
-----------+----------------------+----------
    1_Adeq |        27         11 |        38
    2_Good |        59         28 |        87
  3_Strong |        51          9 |        60
    4_Dist |        36         43 |        79
-----------+----------------------+----------
     Total |       173         91 |       264
```

When doing two-way tabulations, it is a good idea to put the variable with the most categories first so that your table does not wrap. The `miss` option tells Stata you also want to see information on observations with missing data on the tabulated variables. The data we're using for this guide do not have any missing data, so none was returned. However, it is a good idea to use this option when doing your own work.

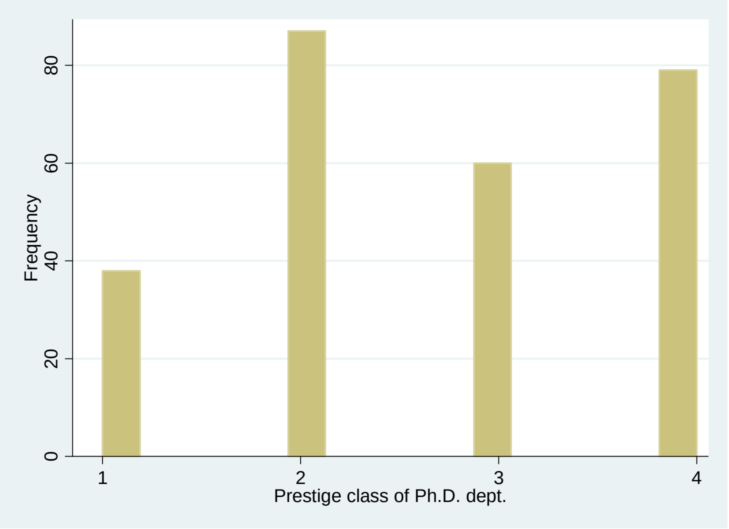
If you want several one-way tabulations, use `tab1`:

```
. tab1 phdclass female, miss
<snip>
```

The help files for `tabulate` are very detailed; we recommend taking a look at them at your convenience. For now, a basic knowledge of the `tabulate` commands is all you need.

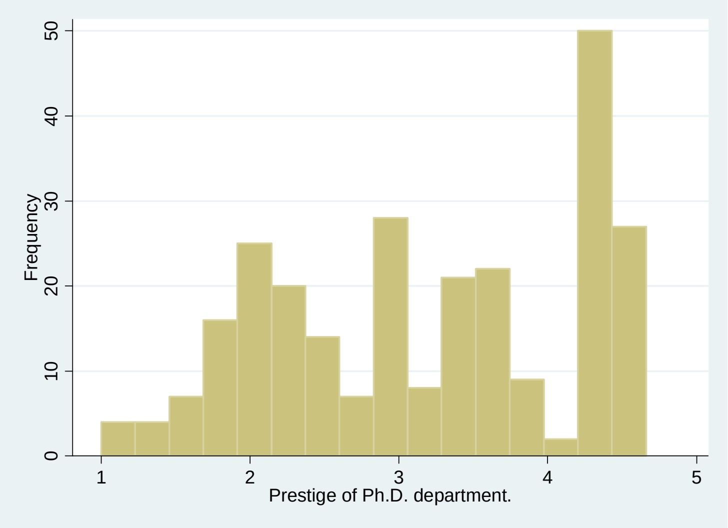
For visual representation of categorical or continuous variables, histograms are a good way to go. The command is very simple:

```
. histogram phdclass, freq
(bin=16, start=1, width=.1875)
```



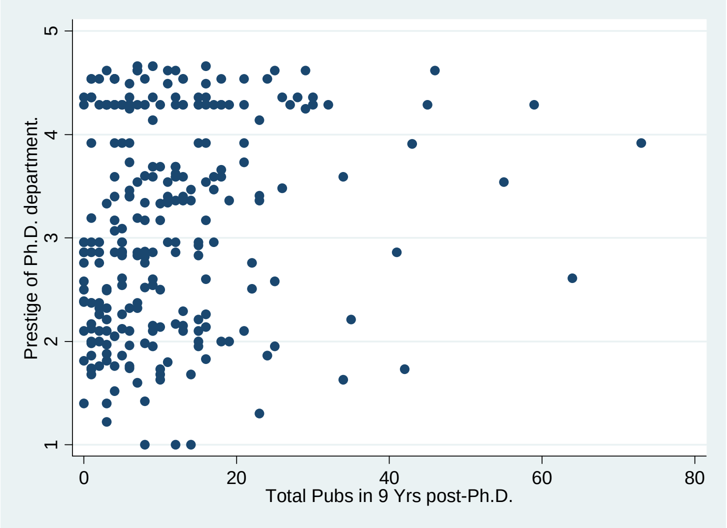
The `freq` option sets the y-axis to represent the frequency of observations. (The percent option is also good.) For continuous variables, the command is the same:

```
. histogram phd, freq
(bin=16, start=1, width=.22874999)
```



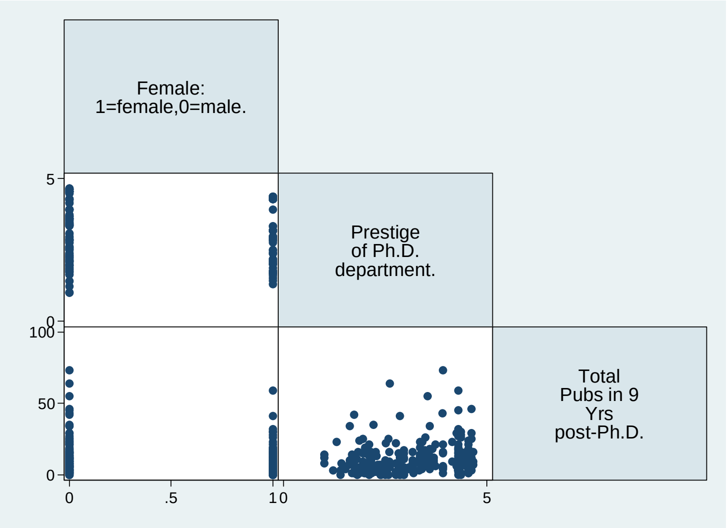
These histograms visualize the information that the `tabulate` command provides. Using the `tabulation` command for continuous variables can produce lengthy output. In fact, Stata will not return output for a two-way tabulation of two continuous variables. In order to see the cross-distribution of two variables, you will need to use the `scatter` command:

```
. twoway scatter phd pubtot
```



You can also look at the cross-distributions of more than two variables at a time. The `scatter` command will only let you do two at a time, but the `graph matrix` command lets you do more. Use the half option to get only the lower half of the matrix (it's a symmetrical matrix, so the top half mirrors the bottom):

```
. graph matrix female phd pubtot, half
```

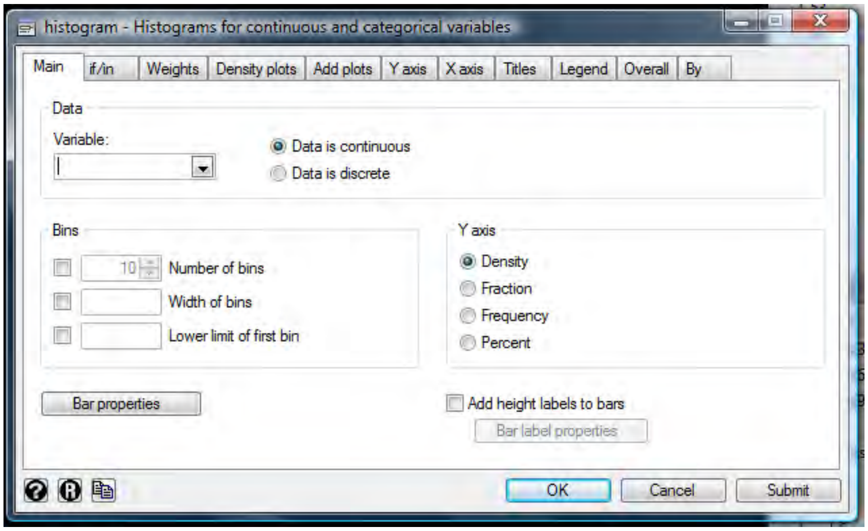


In your assignments for this class, you will want to save your graphs. Here is how you do that:

```
. graph export cda12-stataintro-fig1.png, width(1200) replace
(note: file cda12-stataintro-fig1.png not found)
(file cda12-stataintro-fig1.png written in PNG format)
```

The graph will be saved in your working directory. You can save the graph in many different formats (see `help graph export`); we use the PNG file here. The `width` option determines how big the graph will be. Bigger usually prints better.

One last helpful note on graphs. Later in the course, the options for graphs will become very complex. If you want to try out different options, it might be easier to use the point-and-click features of Stata for graphs. For example, selecting Graphics→Histogram brings up this dialog box:



histogram - Histograms for continuous and categorical variables

Data

Variable:

● Data is continuous

○ Data is discrete

Bins

□ 10 Number of bins

□ Width of bins

□ Lower limit of first bin

Y axis

● Density

○ Fraction

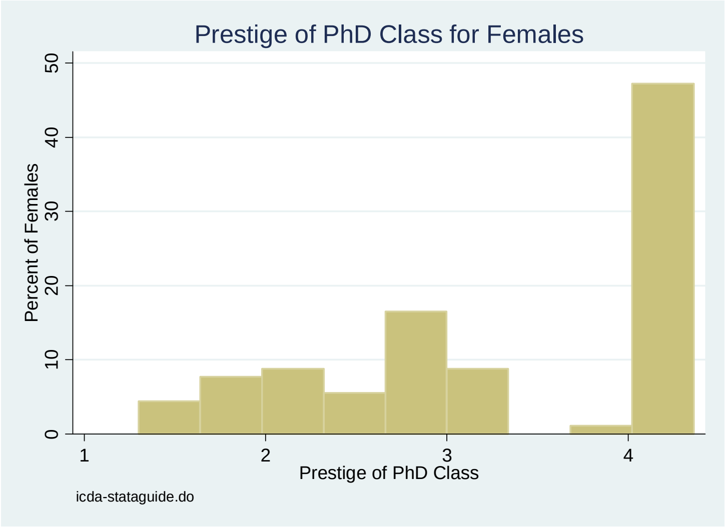
○ Frequency

○ Percent

□ Add height labels to bars

Once you customize the graph the way you want it and submit the command, Stata will return the syntax for that command in the Results Window and produce the graph command and graph:

```
histogram phd if female==1, percent ytitle(Percent of Females) ///
  xtitle(Prestige of PhD Class) title(Prestige of PhD Class for Females) ///
  caption(CDA12-stataintro.do , size(small))
```



You can then copy the command syntax from the Results Window and paste it into your do-file. This way, you'll have a record of the exact commands you wanted (as long as you don't lose the do-file).

Data Management

Creating New Variables

You may want to create new variables or transform existing variables. Here are some examples of how to do this. Each example shows the code for generating the new variable, as well as ways to verify that the transformation is correct. In each example, notice that the commands begin with gen *newvar* =. The command gen is short for generate; you can use either gen or generate.

To create a new variable by adding several others together:

```
      . gen totcit = cit1 + cit3 + cit6 + cit9

      . list cit1 cit3 cit6 cit9 totcit in 1/5

           +-------------------------------------+
           | cit1   cit3   cit6   cit9   totcit |
           |-------------------------------------|
        1. |    0      0      3      9       12 |
        2. |    4      3      8     14       29 |
        3. |    3      1      3     12       19 |
        4. |    0      0      3      9       12 |
        5. |    3      3      8     14       28 |
           +-------------------------------------+
```

To create a new categorical variable from a continuous variable:

```
      . gen phdcat = phd

      . recode phdcat (.=.) (1/1.99=1) (2/2.99=2) (3/3.99=3) (4/5=4)
      (phdcat: 256 changes made)

      . tab phdcat, miss

           phdcat |      Freq.     Percent        Cum.
      ------------+-----------------------------------
                1 |         38       14.39       14.39
                2 |         87       32.95       47.35
                3 |         60       22.73       70.08
                4 |         79       29.92      100.00
      ------------+-----------------------------------
            Total |        264      100.00
```

In the above syntax, the recode command tells Stata that you want observations that were missing for phd to also be missing for phdcat, observations with values 1 through 1.99 for phd will have a value of 1 for phdcat, and so on.

Often it is easier to interpret binary variables than continuous or categorical. The code for creating binary variables is similar to that above:

```
      . gen workres = work
      . recode workres (.=.) (1=0) (2=1) (3=0) (4=1) (5=0)
      (workres: 264 changes made)
      . tab work workres
```

```
   Type of |        workres
first job. |         0          1 |     Total
-----------+----------------------+----------
 1_FacUniv |       141          0 |       141
 2_ResUniv |         0         45 |        45
  3_ColTch |        24          0 |        24
  4_IndRes |         0         33 |        33
   5_Admin |        21          0 |        21
-----------+----------------------+----------
     Total |       186         78 |       264
```

Alternatively, you could use the `replace if` command instead of the `recode` command:

```
replace workres = 1 if work==2 | work==4
replace workres = 0 if work==1 | work==3 | work==5
```

There is also a simpler way to create binary variables:

```
. gen workres2 = (work==2 | work==4) if (work<.)
. tab work workres2

   Type of |       workres2
first job. |         0          1 |     Total
-----------+----------------------+----------
 1_FacUniv |       141          0 |       141
 2_ResUniv |         0         45 |        45
  3_ColTch |        24          0 |        24
  4_IndRes |         0         33 |        33
   5_Admin |        21          0 |        21
-----------+----------------------+----------
     Total |       186         78 |       264
```

The command essentially says: generate a new variable called `workres2`, make it equal to 1 if the variable `work` is equal to 2 or 4 (“or” is indicated by the modulus “|”), and make observations that are missing on `work` also be missing on `workres2`.

Names and Labels

It is not likely that you will need to rename variables in this course; however, this command is used frequently when cleaning new data (e.g., a source variable’s name is V013455). The format for doing so is `rename` *`current-name new-name`*:

```
. rename workres2 research
```

When you rename a variable, everything else about the variable stays the same, including the variable’s label and its value labels. However, when you generate new variables from existing ones, the variable and value labels do not transfer. You’ll want to make sure you attach labels to the variable; otherwise analysis could be very confusing later on.

Next we label the variables we’ve created (Note: in the do-file the variables are labeled immediately after they are generated. This is considered best practice to avoid unlabeled variables).

```
. label var totcit   "Total # of citations"
. label var phdcat   "Phd Prestige: categories"
. label var workres  "Work as a researcher? 1=yes"
. label var workres2 "Work as a researcher? 1=yes"
```

To label values, you'll first need to define the value labels, and then apply them to the variables. Typically, you'll only apply value labels to categorical variables, although sometimes it is helpful to identify what high and low values of continuous variables mean. Here, we've defined and applied value labels to `phdcat`, `workres`, and `workres2`:

```
. label define phdcat 1 "1_Adeq" 2 "2_Good" 3 "3_Strong" 4 "4_Dist"
. label value phdcat phdcat
. label define workres 0 "0_NotRes" 1 "1_Resrchr"
. label value workres workres
. label value workres2 workres
```

As you can see in the first and third lines, you need to define the value label by giving it a name and then specifying what the labels are for each value. As a rule, we name the value labels after the variable to which they are attached. So, the value label for the variable `phdcat` is called `phdcat`. (The exception here is `workres2`, whose value label name is `workres`; since the two variables are the same, it is easier to just define one value label and apply it to both variables.) To check your labeling, you can tabulate the variables:

```
. tab phdcat

        Phd |
  Prestige: |
 categories |      Freq.     Percent        Cum.
------------+-----------------------------------
     1_Adeq |         38       14.39       14.39
     2_Good |         87       32.95       47.35
   3_Strong |         60       22.73       70.08
     4_Dist |         79       29.92      100.00
------------+-----------------------------------
      Total |        264      100.00

. tab workres

  Work as a |
researcher? |
      1=yes |      Freq.     Percent        Cum.
------------+-----------------------------------
   0_NotRes |        186       70.45       70.45
  1_Resrchr |         78       29.55      100.00
------------+-----------------------------------
      Total |        264      100.00

. tab workres2
<snip>
```

If you'd like more information on names and labels, the *Workflow* book has a chapter devoted to this topic.

Beyond the Basics

This section includes features of Stata that will be used later in the course, as well as some techniques that will be handy as your Stata knowledge increases.

Storing estimates and creating tables

You will use the commands `eststo` and `esttab` to display the results from regression models. First, you'll estimate a regression (notice that the first variable in the list is the dependent variable):

```
. logit workfac fellow mcit3 phd

Iteration 0:   log likelihood = -182.37674
Iteration 1:   log likelihood = -164.24112
Iteration 2:   log likelihood = -163.77845
Iteration 3:   log likelihood = -163.77427
Iteration 4:   log likelihood = -163.77427

Logistic regression                               Number of obs   =        264
                                                  LR chi2(3)      =      37.20
                                                  Prob > chi2     =     0.0000
Log likelihood = -163.77427                       Pseudo R2       =     0.1020

------------------------------------------------------------------------------
     workfac |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
      fellow |   1.265773   .2758366     4.59   0.000     .7251437    1.806403
       mcit3 |   .0212656   .0071144     2.99   0.003     .0073216    .0352097
         phd |  -.0439657    .144072    -0.31   0.760    -.3263416    .2384102
       _cons |  -.6344166   .4425034    -1.43   0.152    -1.501707     .232874
------------------------------------------------------------------------------
```

To store the results of this regression:

```
. eststo full
```

Notice that after the `eststo` command, we named this model "full." This is helpful when you go on to compare different models. For instance, you could leave one variable out and compare it to the full model:

```
. logit workfac fellow mcit3

Iteration 0:   log likelihood = -182.37674
Iteration 1:   log likelihood = -164.27165
Iteration 2:   log likelihood =  -163.8246
Iteration 3:   log likelihood = -163.82091
Iteration 4:   log likelihood = -163.82091

Logistic regression                               Number of obs   =        264
                                                  LR chi2(2)      =      37.11
                                                  Prob > chi2     =     0.0000
Log likelihood = -163.82091                       Pseudo R2       =     0.1017

------------------------------------------------------------------------------
     workfac |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
```

```
-------------+----------------------------------------------------------------
      fellow |   1.255574   .2735518     4.59   0.000     .7194224    1.791726
       mcit3 |    .020459   .0065687     3.11   0.002     .0075846    .0333335
       _cons |  -.7544558    .204106    -3.70   0.000    -1.154496   -.3544154
------------------------------------------------------------------------------

. eststo nophd
```

You would then use the `esttab` command and list the models you want in the table. You'll also want to specify model titles, as the default title is the dependent variable.

```
. esttab full nophd, mtitles(Full NoPhD)

--------------------------------------------
                      (1)             (2)   
                     Full           NoPhD   
--------------------------------------------
fellow              1.266***        1.256***
                   (4.59)          (4.59)   

mcit3              0.0213**        0.0205** 
                   (2.99)          (3.11)   

phd               -0.0440                   
                  (-0.31)                   

_cons              -0.634          -0.754***
                  (-1.43)         (-3.70)   
--------------------------------------------
N                     264             264   
--------------------------------------------
t statistics in parentheses
* p<0.05, ** p<0.01, *** p<0.001
```

Finally, you can export the table to a Rich Text Format file, which opens into Microsoft Word. This option produces a publication-ready table, and will make it much easier to present your results. This command

```
. esttab full nophd using cda12-stataintro-table1.rtf, ///
    mtitles(Full NoPhD) replace
(output written to cda12-stataintro-table1.rtf)
```

saves the output to a .rtf file, saving it in your working directory. Here is what the table looks like:

	(1) Full	(2) NoPhD
fellow	1.266***	1.256***
	(4.59)	(4.59)
mcit3	0.0213**	0.0205**
	(2.99)	(3.11)
phd	-0.0440	
	(-0.31)	
_cons	-0.634	-0.754***
	(-1.43)	(-3.70)
N	264	264

t statistics in parentheses
* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$

By default, the `esttab` command returns unstandardized betas. When estimating nonlinear regressions, you may want to include odds ratios in the output instead. To do so, you would need to add exponentiated betas (odds ratios) and standardized exponentiated betas to the saved estimates, and then request them in the table output:

```
. estadd expb: full nophd
. estadd ebsd: full nophd
. esttab full nophd, mtitles(Full NoPhd) cells(expb ebsd) gaps replace

--------------------------------------
                      (1)          (2)
                     Full        NoPhd
                expb/ebsd    expb/ebsd
--------------------------------------
fellow           3.545834     3.509853
                 1.867105     1.857735

mcit3            1.021493      1.02067
                 1.717915     1.683016

phd              .9569868
                 .9567689

_cons            .5302447     .4702664

--------------------------------------
N                     264          264
--------------------------------------
```

As of now, there is not an easy way to export tables with odds ratios into an .rtf file. However, you could easily copy the output from Stata to a Word Document.

Note: The `eststo` and `esttab` commands are very similar to `estimates store` and `estimates table`. You can use either, but be sure to look at the help files for the correct options. You'll need to download the `estout` package to use the `eststo` and `esttab` commands at home.

Using Stata as a Calculator

If you need to do some quick math, you can use Stata's `display` command rather than use a calculator:

```
. display 2+2
4

. di 2^5
32

. di exp(2.915)
18.448812

. di ln(exp(2.915))
2.915
```

The shortcut for `display` is `di`. If you need more information on the operators, expressions, and functions Stata uses, see `help contents_expressions`.

Data Labels and Notes

When saving your data, you may want to attach a label to the dataset. Recall that when we loaded the data used in this exercise, the label appeared below the returned command:

```
. use cleaned-scireview4, clear
(Biochemist data for review - Some data artificially constructed)
```

We've since made changes to the data. You may want to re-label the data to reflect this. Labeling data is much the same as labeling a variable:

```
. label data "Biochemist data for review - Some data artificially constructed"
```

In the label, we've included a brief description of the data so that when we use it, we'll have an idea of what it is.

Also useful are data notes. These are more detailed than data labels, and as such can be longer (data labels are only allowed 80 characters). In these notes, you'd want to include the name of the data, a brief description of what you did, and the name of the do-file you used:

```
. note: cleaned-scireview4-jslv2.dta \ data, added vars totcit, phdcat, ///
> workres, and workres2 \ scott long cda12-stataintro.do 2011-08-25.
```

You can also attach notes to variables. If you create new variables from existing variables, as we did above, it is helpful to keep a record of the new variable's source:

```
. note totcit: created by adding cit1 cit3 cit6 cit9 ///
> \ scott long cda12-stataintro.do 2011-08-25.
```

The dataset name we wrote in the data note is not the same as the current data we're using. Since we have changed the data, we will want to save it with a new name. The name of the new dataset is indicated in the note. To save the revised dataset:

```
. save cleaned-scireview4-jslv2, replace
(note: file cleaned-scireview4-jslv2.dta not found)
file cleaned-scireview4-jslv2.dta saved
```

To see the label and notes you've created:

```
. use cleaned-scireview4-jslv2, clear
(Biochemist data - updated for getting started review)

. notes _dta

_dta:
  1.  add labels to sci.dta and add recoded variables \ jsl 1998-05-24.
  2.  science.dta \ merge mysci and sciplus \ jsl 2000-05-03.
  <snip>
10.  cleaned-scireview4-jslv2.dta \ data, added vars totcit, phdcat, workres,
      and workres2 \ scott long cda12-stataintro.do 2011-08-25.

. note totcit

totcit:
  1.  created by adding cit1 cit3 cit6 cit9 \ scott long cda12-stataintro.do
      2011-08-25.
```

Locals

Chapter 4 in *Workflow* details the use of local macros for automating your work. Locals are analogous to a handle, where you designate an abbreviation to represent a string of text. Locals can be used as tags:

```
. local tag "CDA2011-stataintro.do slr 2009-05-19"
. note workres: created from work \ `tag'.
. note workres

workres:
  1.  created from work \ CDA2011-stataintro.do slr 2009-05-19.
```

In this example, I called my tag tag, and am telling Stata that I want tag to stand for what's inside the quotation marks. When I create notes for my variables or data, I can quickly type `tag’ to stand for the do-file name, my initials, and the date. Notice that the opening single quote is different from the closing single quote. The opening quote is found above the Tab button on your keyboard (on the same key as the tilde (~)), while the closing quote is the standard single quote (to the left of the Enter button).

Locals are also used to hold lists of variables. For instance, you can use a local macro to represent the right-hand-side (predictor) variables:

```
. local rhs "workfac enrol phd"
. regress pubtot `rhs'

      Source |       SS       df       MS              Number of obs =     264
-------------+------------------------------           F(  3,   260) =   10.77
       Model |  3519.43579     3  1173.14526           Prob > F      =  0.0000
```

```
    Residual |  28326.1968   260  108.946911           R-squared     =  0.1105
-------------+------------------------------           Adj R-squared =  0.1003
       Total |  31845.6326   263  121.086055           Root MSE      =  10.438

------------------------------------------------------------------------------
      pubtot |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
     workfac |   5.227261   1.297375     4.03   0.000     2.672561     7.78196
       enrol |  -1.174879   .4465778    -2.63   0.009    -2.054249   -.2955094
         phd |   1.506904   .6442493     2.34   0.020     .2382931    2.775514
       _cons |   9.982767    3.33341     2.99   0.003     3.418849    16.54668
------------------------------------------------------------------------------
```

If you use the same variables several times throughout your do-file, you can simply type `` `rhs’ `` instead of the whole variable list. Additionally, if you need to change the variable list, you will only need to change it once—in the local.

At the end of your do-file, don’t forget to close the log file. If you don’t, any work you do after running this do-file will be recorded in `cda2011-stataintro.log`. Then, make sure there is a hard return after the log close command. The easiest way to remember to do this is to type `exit`. The `exit` command tells Stata not to read any further in the do-file:

```
. log close
      log:  E:\CDA stata guide\CDA2011-stataintro.log
  log type:  text
 closed on:  8 May 2009, 11:04:25
-------------------------------------------------------------------------------

. exit

end of do-file
```

Provenance: 2009-05-19; 2011-06-16; cda12 StataGettingStarted 2011-08-27.docx; cda12 StataGettingStarted 2012-05-11.docx