

Introduction to Stata

Getting Started with Stata Programming

Nicholas P. Nicoletti, Ph.D.

Missouri Southern State University
International and Political Affairs

June 24, 2019

Abstract

This document is intended as a beginners guide to research with Stata. It was originally developed for the University at Buffalo (SUNY) Political Science Department PSC 531 Lab. This guide may be used in conjunction with the referenced .do files and data sets.

Contents

1	Introduction: What is Stata?	3
1.1	The Basics	3
1.2	The Importance of Logging	4
1.2.1	Directories	5
1.2.2	Creating and Maintaining Log Files	6
1.3	“Programming on the Fly” vs. Do-Files	7
1.4	Opening and Saving Stata (.dta) Files	8
1.4.1	Importing Data	10
1.4.2	Memory	13
1.5	Do-Files	14
1.6	Help	15
1.7	Plug-ins	17
2	Command Syntax	17
2.0.1	Basic Command and Operators	19
3	Working with Data	21
3.1	Building Datasets from ASCII Text Files Using Do-Files and Dictionary Files	21
3.2	Labeling Variables	26
3.3	Summary Statistics and Histograms	27
3.4	Generate and Recode	28
4	Hypothesis Testing	31
5	Introduction to Correlation and Regression	35
5.1	Simple Post Estimation Commands	36
6	Advanced Regression Models	39

7	Limited Dependent Variables Regression Models	40
7.1	Logistic Regression	41
7.2	Logit Example	42
7.3	Probit Example	46
8	Graphing in Stata: A Simple Example	47

1 Introduction: What is Stata?

Stata is a statistical software package used by scholars in many fields; it is most common in the Social Sciences such as Political Science, Economics, and Sociology (Teele, 2010). Stata is primarily run from a “command prompt”, although users can employ the “drop-down” menus to perform most of the common types of analysis. This introduction will focus on using Stata in the command prompt form. The basic structure of any dataset is a series of rows and columns (i.e. matrices). Users manipulate data with various commands which transform raw data into meaningful statistics which can be interpreted by the researcher. Stata, although similar to Microsoft Excel, is much more powerful and allows the user to store all commands in a log file — often called a .do file, after its program extension. Stata can generate tables, graphs, and be used to apply various statistical models. The program also integrates well with the typesetting program L^AT_EX for the seamless creation of stylish output/results. On a quick note, throughout the document all Stata code will be placed between `||` — as if they were absolute values. This is so you can easily identify Stata code in the text of this document. *You do not need to include the `||` in your Stata code, it is only there to make the commands easier to read in the document’s text.*

1.1 The Basics

The Stata display will contain four primary windows:

1. Command Window
2. Results Window
3. Review Window
4. Variable Window

The command window is where the user enters commands for Stata to execute. Simply type a command and hit enter. The results of the command will appear in the Results Window; this pane will display the entered command, followed by its corresponding

output/results. The content can be copied and pasted into other editors, but cannot be edited on the screen. Although one can copy and paste a table from Stata's Results Window directly into word, it is recommended that you refrain from doing so. The variable names will be hard to interpret for those who are not familiar with your codebook and none of the information will be conveniently displayed. Your audience (especially your professors and reviewers) will not want to interpret raw Stata output. Later in this guide I will show you how to take most of your Stata output/results and seamlessly plant it into a \LaTeX document. If you are not using \LaTeX create stylish tables in Microsoft Excel or Word. The Review Window displays a list of commands that you have already completed. You can easily rerun a command you have already completed with a few steps. First, highlight a command in the Review Window, the command will then appear in the Command Window. Now you can click enter and run the command. Finally, the Variable window will display the list of all variables in your currently loaded dataset. This window is empty when you first start Stata and contains content after you have uploaded data into memory.

1.2 The Importance of Logging

Every time you begin a new project you should begin a new **Log** file dedicated to that project. The log file captures all the printed text in the Results Window — this includes the commands you have typed and the output/results Stata has displayed. Of course it will also display all of your errors and failed attempts. A log file is important to research with Stata because often times you will forget many of the commands that you ran in your last session. You may not remember the different regression models you tried with different variable sets. You might have used a command for the first time last session that you do not remember now. If you have a log file, this will not be a problem for you because you will be able to reference your sheet and recreate your output/results.

1.2.1 Directories

Before we look at logging it is important to outline **Directories** in Stata. A directory is just a folder somewhere in your computer. For example, usually your hard drive is called the C:\ drive. Directories help to keep your folder organized; obviously you do not keep all of your files in the same place on your computer (occasionally you will meet the user that keeps everything on their desktop). Stata has a default directory within its folder on the C:\ drive. You will want to change the **Directory Path**, which is the series of folders where you want your files stored. There are two primary commands you will need when dealing with Directories. First, you will need the **cd** or “change directory” command. This command will allow you to change where Stata stores the documents you will be using, including the log file which we will cover in a moment.


I am using my flash drive to store all the documents associated with this document — this drive, on my computer (it would be different on yours), is located in the G:\ directory. To change the directory the command would be: | **cd G:** |. However, I do not want Stata to use the entire G:\ to store files; I want to place my files into a folder. To do this I simply specify the full directory path. The code would be: | **cd G:\Log** |. This will change the Stata directory to a folder called Log on by G:\ drive. This is important for log files because whatever directory you are in, is where the log file will be stored — as well as other Stata files.

The second command you should be associated with with is **dir**. If you are not familiar with what folders are available to you in the current directory you can type **dir** and it will list all the folders in the current directory. You can then use the **cd** option to change the directory to the desired location. Now we can move on to creating a log file.

1.2.2 Creating and Maintaining Log Files

To create and use log files you will need the following commands:

- | **log using** *my log file name* | — This will tell Stata to open a log file which will record everything you type in the command window and output you see on the screen. It will also be necessary to tell Stata in what directory to place your log file (see above).
- | **log close** | — This will turn logging off.
- | **log using** *my log file name, replace* | — This will tell Stata to overwrite the existing log file you are using.
- | **log using** *my log file name, append* | — This will tell Stata to append (add on to) an existing log file (recommended when continuing a current project).
- | **log off** | — Temporarily stops logging.
- | **log on** | — Resumes logging.

On a quick note, log files can also be created using the Graphical User Interface (GUI) menu in Stata 11. Using the “Log/Begin/Close/Suspend/Resume”  button on the tool bar, you will be able to create a log file, choose which directory to place your file, the name of your file, and whether to overwrite or append existing files — just as you would in any PC or MAC-Based GUI.

Logs can be edited later in a text editor such as Notepad or Wordpad. However, to make the log file readable in these programs we must change it to a .txt file — otherwise it will remain the default Stata .smcl file. To do this you will use the command: | **translate** *my log file name.smcl log file name.txt* |. Now you will be able to edit the file in a text editor and also create a Do-File from its contents. Do-Files will be discussed further later, but this is a good time to make a vital point about “programming on the fly” vs. Do-Files.

1.3 “Programming on the Fly” vs. Do-Files

“Programming on the Fly” is a common term used to describe when a user types commands into Stata’s command prompt without “running” them from a .do file. Programming on the fly is useful when one is playing with the data. Many times you will make errors and Stata will not be able to execute the botched command. However, when working in Stata it is **strongly** recommended you use a .do file. When you have run a command that is useful you can easily export it into a .do file. Recall that all other commands will be stored in your log file and can be exported into a Do-File later.

Stata will continually show all of your commands in the **Review** box in the upper-left hand side of the screen (see image below). Simply right click on the command you want to export to the .do editor and then click on “Send to Do-File Editor”. This action will open a new .do file editor (if one is not already open) and place your command on the next open line. Figure 1 shows the Review box, Figure 2 shows the the drop-down menu, and Figure 3 shows Stata’s Do-File Editor.

The command displayed is called | **set mem** |. Sometimes the default memory Stata allocates is not enough to use larger datasets. The **set mem** command allows the user to change how much memory Stata allocates to your data. Typing “set mem 500m” sets the usable memory to 500 MB, which is usually sufficient for large datasets. The | **set mem, perm** | command allows the user to permanently set the memory to a desired allocation.

Do-Files are a very important part of the Stata experience. Saving all commands used to manipulate data or make a calculation will make it easy to reproduce your results very quickly in the future. Creating .do and log files are also a great tool for students; sometimes you will run into research problems — maybe with Stata commands or with your statistical model — a log or .do file will easily allow you to show your work to a more experienced scholar which can then assist you with your issue. Do-files will be

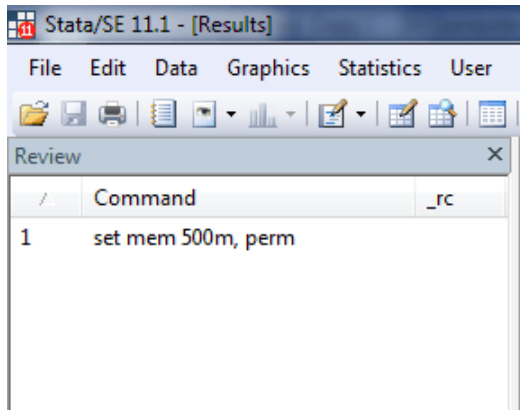


Figure 1: Review Box

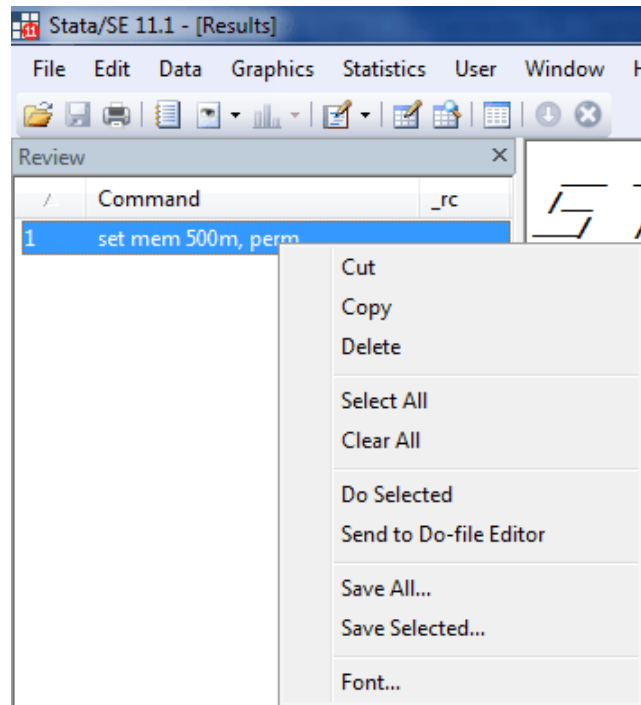


Figure 2: Right-Click Drop-Down

discussed more later, but the major point of this section is that log files and .do files are a necessary and important part of research with Stata.

1.4 Opening and Saving Stata (.dta) Files

The final section of this part contains what you need to know about data files and Stata. First, Stata is a great data/variable manipulation tool; however, it is not always the best program to use when compiling your data. Many times it is preferable to use a program like Microsoft Excel to contain and compile your dataset. The Stata data editor is limited in many ways to what it can perform. Nevertheless, it is strongly recommended that you keep a copy of your un-manipulated raw data file. In the course of manipulating data in Stata you will change and transform your variables. Many of these changes cannot be undone! Having a copy of untouched raw data will be very advantageous when you need to restore a variable that was on the interval scale until you transformed it into a dummy variable with Stata (this, or something like it, *will* happen to you at some point in

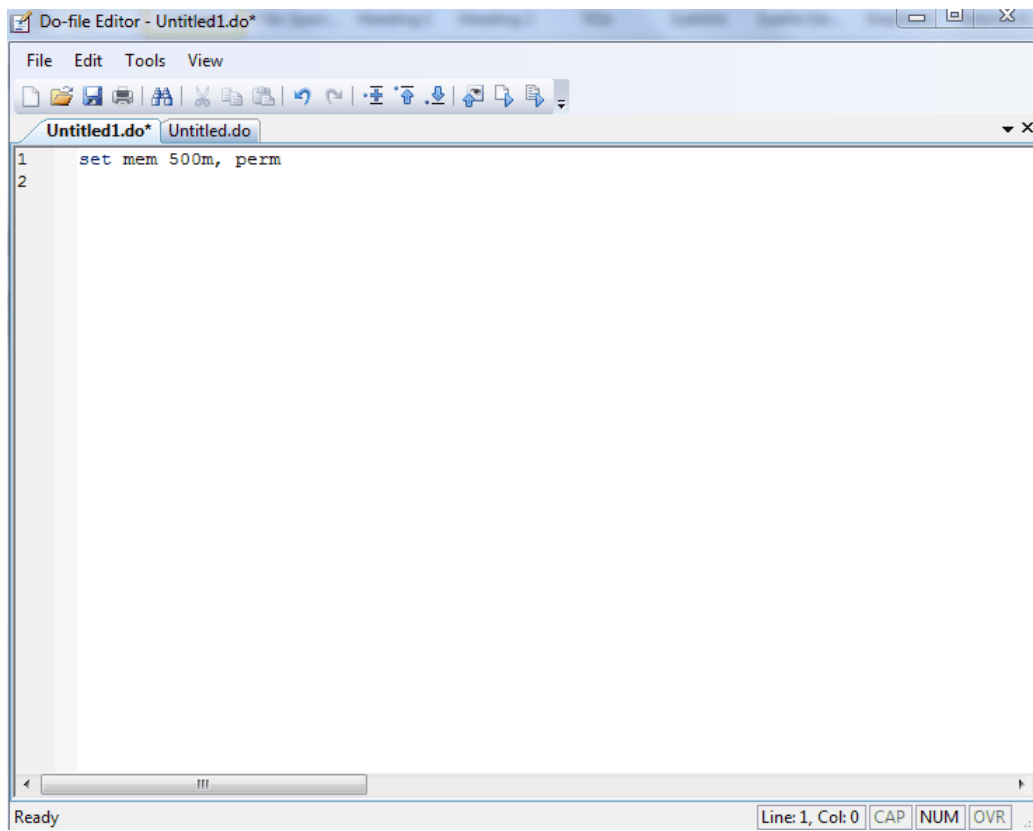


Figure 3: Do-File Editor

time — save yourself, keep a backup!).

Raw data files can be contained in many different types of files — these are all essentially text files under the The American Standard Code for Information Interchange (ASCII - pronounced “ask-ee”). The most common for data retention are the .csv (comma separated values or comma delineated values) and simple tab delineated text files. I use .csv files because Microsoft Excel can read/save these files, while also allow a user to manipulate the data using standard Excel commands. I recommend that you begin with .csv files when compiling your data and also save all raw data files in this format. Stata has no problem importing .csv files and they are also compatible with all other statistical packages (i.e. SPSS, Minitab, SAS, etc.). Moreover, while Stata 11 can read all earlier versions .dta files, the same is not true for earlier versions. For example Stata 8 cannot read a Stata 11 .dta file — you will receive an error. But all versions of Stata can read .csv files. Stata 11 can also save .dta file in older formats if necessary using the command: | **saveold** *my file name* |.


1.4.1 Importing Data

Data from a any spreadsheet (.xls, .csv, etc.) can easily be imported into Stata. If your data comes from another stats program file you will have to convert it to a .csv or .dta file first. If your data is in the default Excel .xls format convert it to .csv first. Also make sure to avoid placing spaces in your variable names and making them too long — Stata may have trouble importing the variable names otherwise. For example, if I have a variable named GDP Per Capita, I may rename it *gdppercap* — but remember to change the variable name back when presenting the output/results to an audience. Thus, variable names should contain no spaces and should be located in a single row separating the data from the variable names.

To import the .csv files you will use the | **insheet** | command which transfers the spreadsheet file into Stata. You will need to specify the entire directory path from Do-Files, but if you are programming on the fly, and you told Stata what directory you want to be in, and your data is in that directory, you can specify the name of the file only. Here is an example of the code:

- | **insheet using** *my file name.csv*, comma | — use the comma specification if it is a .csv otherwise you do not need this addition. You must specify the file extension at the end of the file name — in this case it is a .csv file.
- | **insheet using** *my file name with full directory path.csv*, comma | — you will need to specify the full directory path if you have not set up your directory properly in Stata.

Certain things in Stata can be done quicker using the GUI — especially if your directory path is really long and contains multiple characters. You can simply click File, Open (or use the folder icon on the toolbar) and search for the directory your data is in and open it from there. However, when compiling a Do-File and using log files, telling Stata what directory it should work from synchronizes your project so everything is organized in a preferable way.

There is another way to import data into Stata — I tend to use this way the most and find it to be the most efficient, although I still recommend that you set Stata in the proper directory. First, have your spreadsheet open. Second, click on the data editor button  in the Stata toolbar. The Stata Data Editor will open up and you can view it in Figure 4. Copy the entire contents of your spreadsheet and paste them into the first cell of Stata's data editor. Stata will then ask you if the first row of the pasted data contain variable names, if it does click the appropriate box which is depicted in Figure 5.

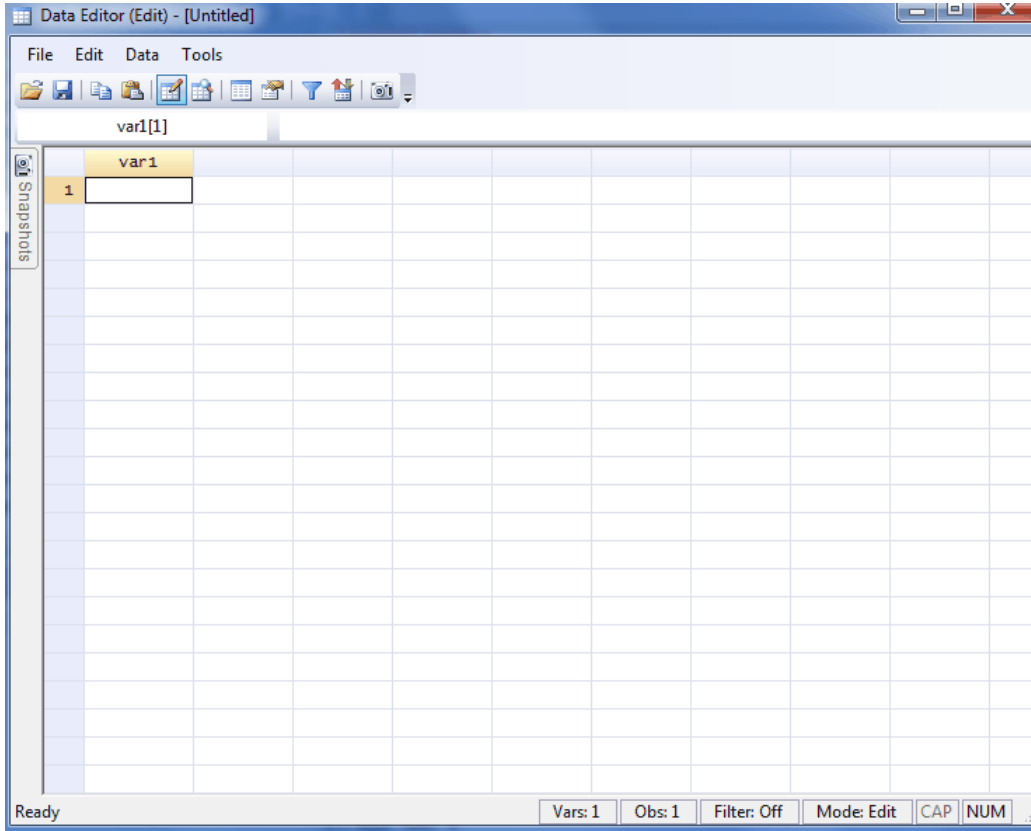


Figure 4: Stata Data Editor

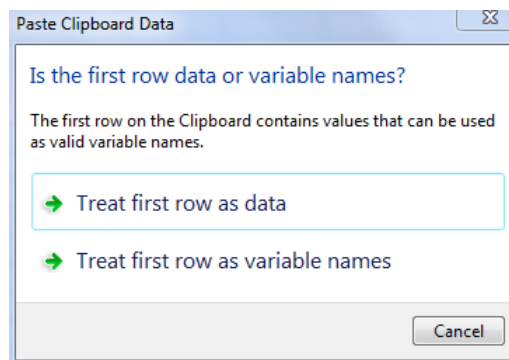


Figure 5: First Row Variable Prompt

When you close the data editor Stata will load your variables into memory and they will appear in the Variables Window. You are not finished yet. Now you need to save your data as a Stata .dta file. As with everything in Stata this can be done using command line and GUI. To use the command line simply type | **save my file name** | and Stata will save the file in the directory you have specified (see above). You can also use the GUI interface by clicking File, Save or the save icon on the toolbar. You can then choose which directory you wish to save the file to. Just as you save a raw copy of your data in a .csv file, it is also recommended that you save a second copy of your .dta file — just in case. Teele (2010) recommends saving a primary file and a “working file”. Your working file is what you will use to manipulate data during your project and your “original file” is your untouched backup. For example use the commands:



- | **save my file name-original** | — This is the untouched .dta file.
- | **save my file name-working** | — This is the file you will use.
- | **save file name-working, replace** | — The replace command will overwrite the working file you have been manipulating.
- | **saveold** | — Saves the file in older versions of Stata.

1.4.2 Memory

Sometimes you will need to increase the amount of memory Stata allocates to your data. For example the American National Election Study (ANES) dataset is far to large for the default Stata memory allocation of 10MB. Below are the commands necessary to set Stata’s memory allocation.

- | **set mem number of bytes m** | — Sets the memory to your choosing in megabytes.
- | **set mem number of bytes m, perm** | — Sets memory to your choosing in megabytes permanently.
- | *Example:* **set mem 500m** | — Set the memory allocation to 500 MB.

1.5 Do-Files

A Do-File is a Stata file with the program extension `.do` (I have called them `.do` files or Do-Files). These contain a set of commands that can be read and run by Stata. You can easily create a new Do-File by clicking on the “New Do-File Editor” icon  button. This will open up the Do-File Editor depicted in Figure 3. You can also use the command `| doedit` in the command prompt window. When you have lines of command in a Do-File you can highlight these commands and click on the “Execute (do)”  icon in the Do File Editor toolbar. Stata will then execute your command and present your results as you typed the command into the command line window. You can also create Do-File in a text editor such as WinEdt or Notepad. As noted above in Figures 1 and 2, you can also send commands from the Review Window to the Do-Editor by right clicking on the command and then clicking “Send to Do-File Editor”.


You can also annotate a Do-File or make notes within a Do-File by opening and closing a set of asterisks (*). For example, **Note: The above command is for a pooled cross sectional time series dataset**. Notice how I opened and closed the set of *. Stata will ignore lines with an * in front, but you also need to close it with an * or Stata will ignore the rest of your Do-File.

Do-Files STOP when there is an error in the code. Sometimes we have to tell Stata to ignore an error in our code. What if we need to tell Stata to drop a variable as it runs the analysis. This can be done with the **capture** command. For example writing, `| capture drop variable-name` within your `.do` file allows Stata to drop the variable if it exists or ignore the command if there is an error.

The Do-File Editor also allows us to tell Stata not to execute a command until it sees a certain punctuation — this is called a delimiter. The delimiter command can be used

when we want to pause a Do-File or have the line of command span more than one line. To do this use the `| #delimit ; |` command — here we are telling Stata not to execute a command until it sees the `;` punctuation. You can also turn the delimit command on and off (which is really important because you will not want to delimit everything); this can be done using the `| #delimit cr |` command. This will stop the need to use the punctuation to continue to executer commands. Make sure to use this command otherwise Stata will keep scrolling through the Do-File without executing any other commands that come after.

1.6 Help

Stata has a decent set of help files built into the program. To use the Stata 11 help files type: `| help command-name |`. Stata will open a new window with all the help files associated with the command you typed in. For example, typing `| help reg |` will open the widow depicted in Figure 6. The help file tells the user that the “**reg**” command performs a linear regression and then lists the exact syntax of the command. In this case, to use the **reg** command a user must specify: `| reg dependent variable (depvar) [set of independent variables (indepvars)] [if] [in] [weight] [,options] |`. The help window then explains each of the different options with links to the dedicated help page of each one. Lastly, the help file lists commands associated with the command the user needed help with; in this case Stata lists different types of regression commands, such as logit, probit, and tobit — which we will become familiarized with later. If the user is not familiar with the command they need assistance with they can search through the help files once Stata opens the help dialog box; to do this simply type: `| help |` and use the search box  to find the commands associated with the operation you want to perform.

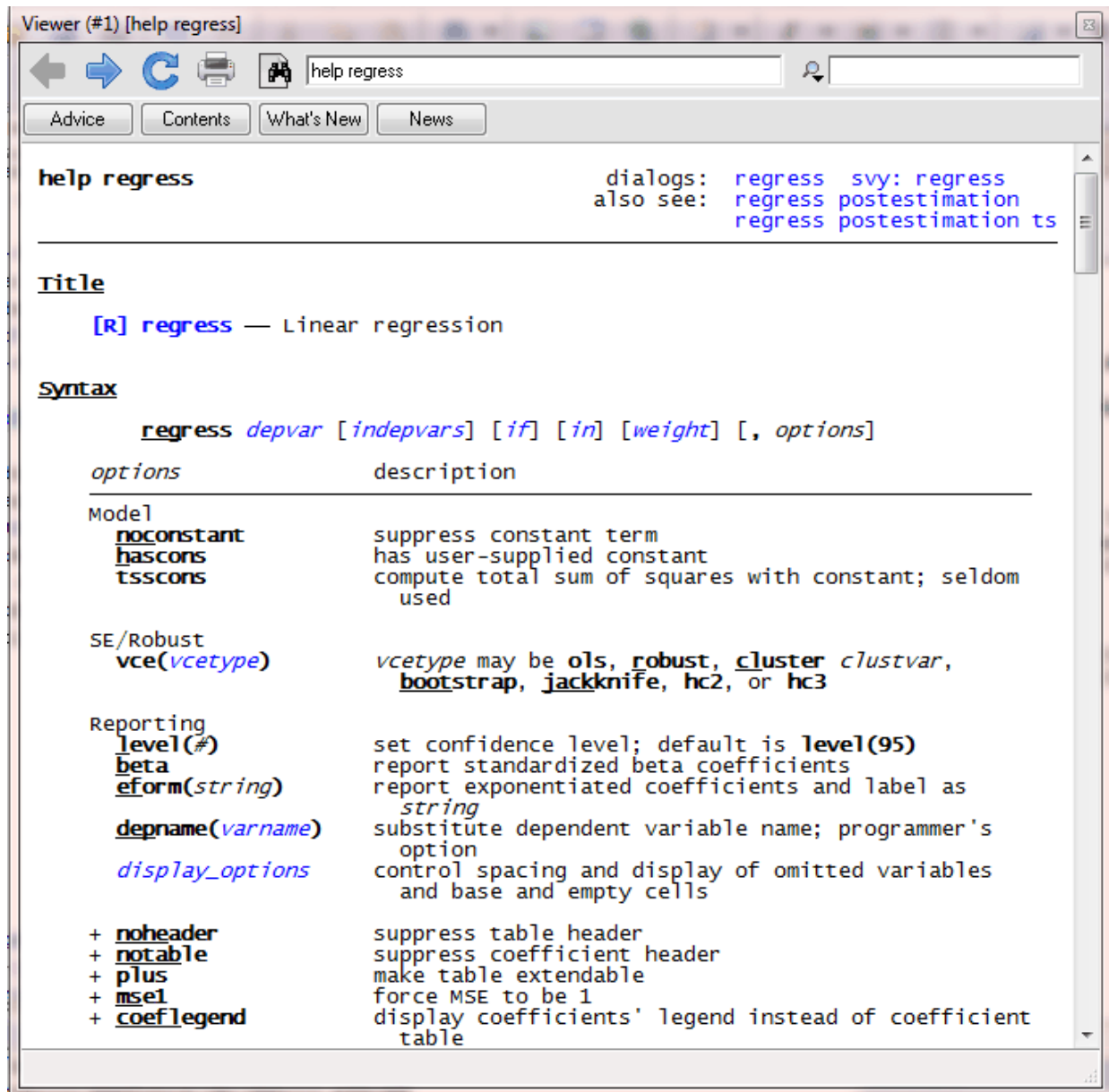


Figure 6: Help Window

1.7 Plug-ins

Stata is not “open source” software such as the free statistical package **R**; however, often times scholars write code for use with Stata which can make difficult operations easier. For example, probit and logit coefficients cannot be interpreted the way that Ordinary Least Squares (OLS) linear regression coefficients can; whereas OLS unstandardized coefficients can be interpreted as a one unit increase in X is associated with a coefficient sized increase (decrease) in Y , logit and probit coefficients must be converted into predicted probabilities. Tomz, Wittenberg and King (2003), from Harvard University, have developed a set of Stata commands contained in the plug-in Clarify, which allows users to easily generate predicted probabilities with a few simple commands. Finding and installing these Stata plug-ins is easy. Use the command: | **findit** *program name* |. For example, to find and install Clarify one would type: | **findit** clarify |. Stata will then open up a new window containing all of the associated Stata internet files that contain the word “clarify”. Notice in Figure 7 that the Clarify plug-in is second package listed. The user will then click on the Clarify package to bring up another window which contains an “install” link. Clicking on “install” will prompt Stata to automatically install the package.

A common place to find useful Stata plug-ins is the Social Science Research Council (SSRC). Type the command | **ssc install** *program name* | to install a program from the SSRC website. If you do not know which program you want to install type the command: | **ssc hot** | to pull up a list of SSRC Stata plug-ins. Click on each program for a description of its use. To install the packages follow the same instructions as above.

2 Command Syntax

Thus, far we have been using Stata command syntax for various operations, such as changing directories and importing .csv files. This section will explain more on the proper

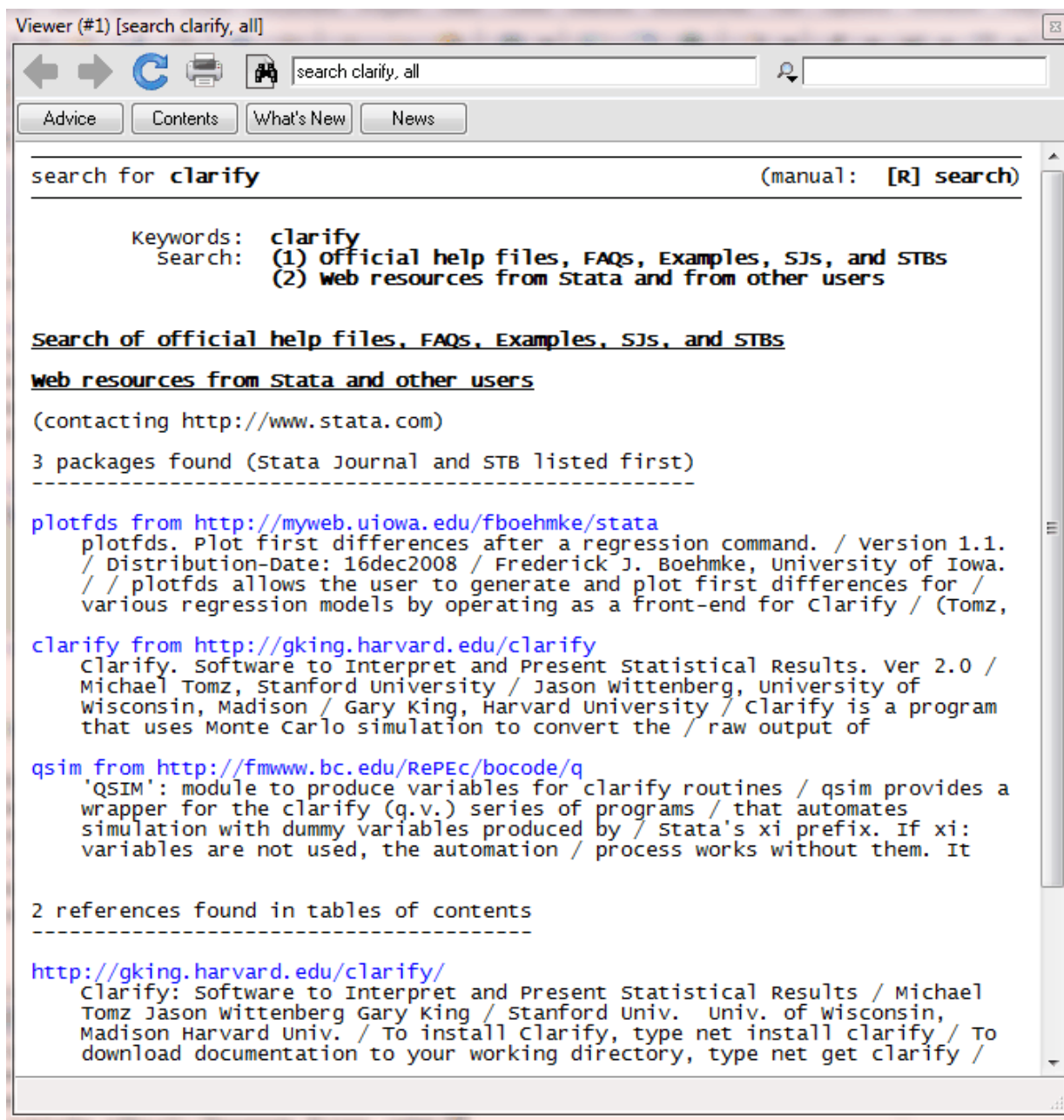


Figure 7: Findit-Clarify Window

Stata syntax and teach you how to use Stata to begin manipulating data. All Stata commands must be written in proper **syntax** to be executed. Syntax is the sentence structure or *language* that Stata understands. Stata syntax is structured in the following way: | **command** arguments, options |. The “command” is the name of the command you want Stata to execute. The “arguments” are things like variables that you want the command to execute an operation over. The “options” are additional pieces of information that you can give to Stata in order to execute ancillary operations. The “options” must always come after a “,” and in general there is only one comma per command (Teele, 2010).

We have already covered several examples of Stata syntax. For example, typing: | **help reg** | tells Stata that you want it to open up the help window containing the help file on the *reg* command. As stated earlier, the help command is perfect for finding the proper Stata syntax for any command. Reference Figure 6. At the top of the file you will see the heading **Syntax**. Under this heading is the proper Stata language for executing the “reg” command. The next section lists the relevant “options” associated with the “reg” command.

2.0.1 Basic Command and Operators

Stata works with two different types of variables: (1) numeric variables (continuous, interval, categorical/dichotomous) and (2) string variables (combination of alphabetic and/or numeric). In order to manipulate these variables and use the commands in Stata it is important to know what **operators** Stata uses, as intuition is not always what you would think.

Operators:

- + Addition
- - Substraction
- * Multiplication

- / Division
- ^ Raise to a power
- > Greater than
- < Less than
- >= Greater than or equal to
- <= Less than or equal to
- == Equal to (the operator for equality is a pair of equal signs)
- ~= or ! Not equal to or
- & and
- | or
- ~ Not
- abs(x) Absolute value
- exp(x) Exponent
- ln(x) Natural logarithm
- log(x) Natural logarithm
- log10(x) Logarithm to the base of 10
- sqrt(x) Square root

Basic Mathematical Operation

- | display 2+2 |
- | di sqrt(2)/2 |
- | di 4*4 |

Reference the “Basic Mathematical Operation” list above. This list contains the **display** command. The “display” command tells Stata to display strings and values of scalar expressions. Notice that the first two letters of the command are underlined. If one were to open up the help file associated with the display command they would find that the first two letters of the command are also underlined. This indicates that the command

can be abbreviated when programming on the fly. For example, typing | **di** 2+2 | would be the same as typing | **display** 2+2 |. Many Stata commands have a similar shortcut. For example the Stata command | **generate** |, used to create a new variable, can be abbreviated using | **gen** |. All of the Stata abbreviations can be found within the command’s help file.

It might also be helpful to know that Stata allows commands to run both “noisily” and “quietly”. The default in Stata is to show you the output/results of your commands - this is known as running commands “noisily”. However, for example, if you are going to use a regression output in another regression (as in an instrumental variables approach) you may want Stata to skip its output/results for the first regression. This is mainly used in conjunction with Do-File operations. To do this simply type **quietly** in front of your primary command. For example, | **quietly tabulate** V083210 |, which will tabulate the ANES gender variable and store it in memory without presenting the output/results.

3 Working with Data

Stata handles a combination of numeric and string variables. In the Variables window you will see the *name*, *label*, and *format* of each variable. The *name* is a limited variable name that we will use to tell Stata what variables to perform operations on. The *label* gives a more detailed description of each variable. The *format* tells us whether the variable is a string or a numeric variable — the string variable format will be followed by an **s**, while the numeric variable format will be followed by a **g**.

3.1 Building Datasets from ASCII Text Files Using Do-Files and Dictionary Files

This section will begin to look at how to manipulate data. One of the most important skills to have when working with Stata is knowing how to use various Do-Files to create

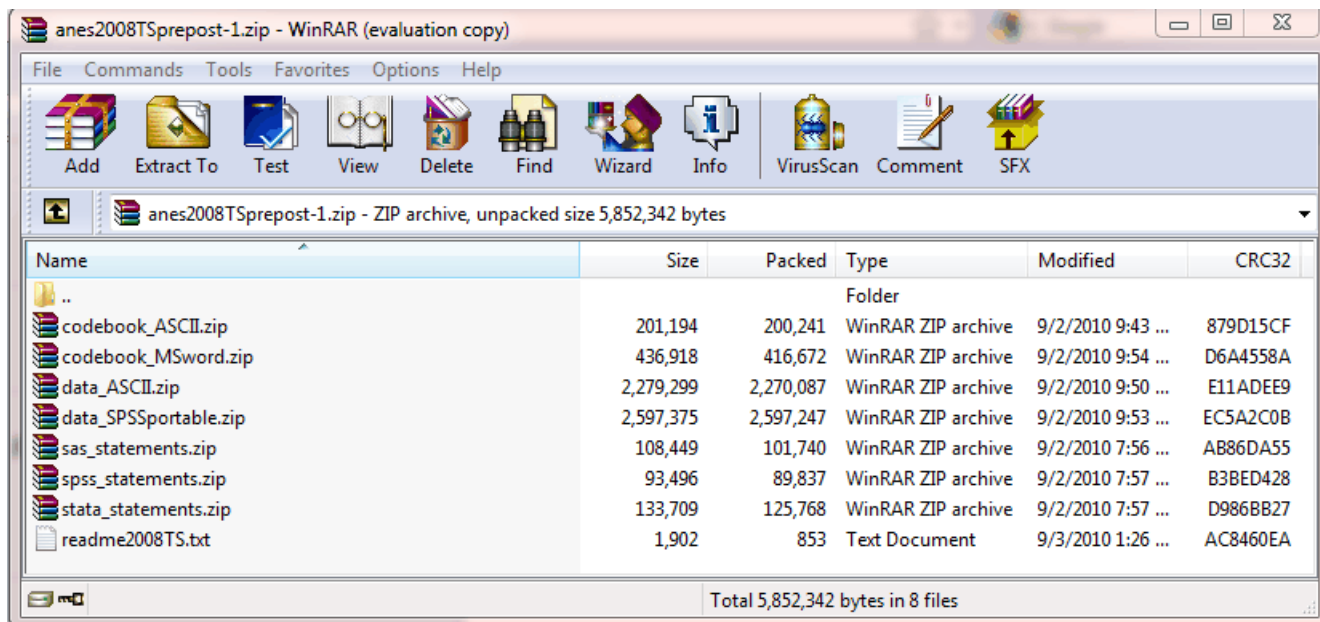


Figure 8: 2008 ANES Time Series .zip

Stata data sets. Sometimes (more times than not) when data is posted online, it is posted using an ASCII .txt file and a universal dictionary file with the extension “.dct”. This is done so that a single text file can be used to create data sets in multiple statistical packages (SPSS, SAS, Stata, Minitab, etc.). I am going to use an example from the American National Election Study (ANES) and use the ANES data to demonstrate several other commands in Stata. You can follow all the steps in this section to practice building data sets from Do-Files.

First go to the American National Election Study (ANES) website at <http://www.electionstudies.org/>. Click on Data Center. I am going to use the 2008 ANES Time Series Data for this demonstration. Click on *download.zip file (all)* and proceed to download the .zip file containing all the files needed to build the data set. Figure 8 shows what is inside the .zip file (I use the WinRAR compression program to unzip the files).

Notice how there are several .zip files within the original .zip file. Figure 9 shows the

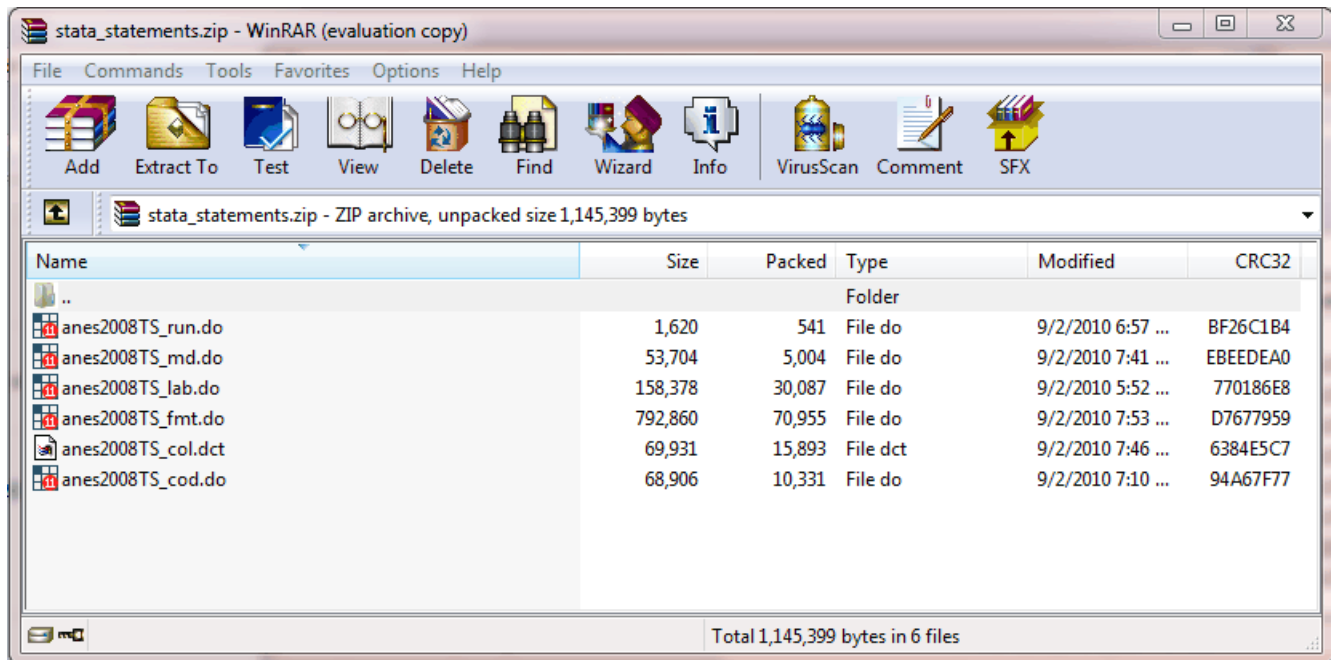


Figure 9: 2008 ANES Time Series Stata .zip File

files contained in the Stata.zip file. Figure 10 shows the contents of the Data_ASCII.zip file, which is the text file we will use to create our .dta file.

The next step is to unzip the necessary files to a directory on your computer. Initially the files can be unzipped to any directory you want; however, we will need to change the directory so that Stata can execute the Do-File. Open the Do-File called:

anes2008TS_run

This is the Do-File necessary to execute the command Stata will need to turn the raw text and dictionary file into a .dta file. Figure 11 shows the contents of this Do-File. This file tells the user the default directory that the Do-File requires all the files to be placed in order for Stata to execute the commands and build the data set.

Most of the time, you will need to place the files within the C:\ directory — remember the “\” character indicates a new folder. To use this data you will need to create the

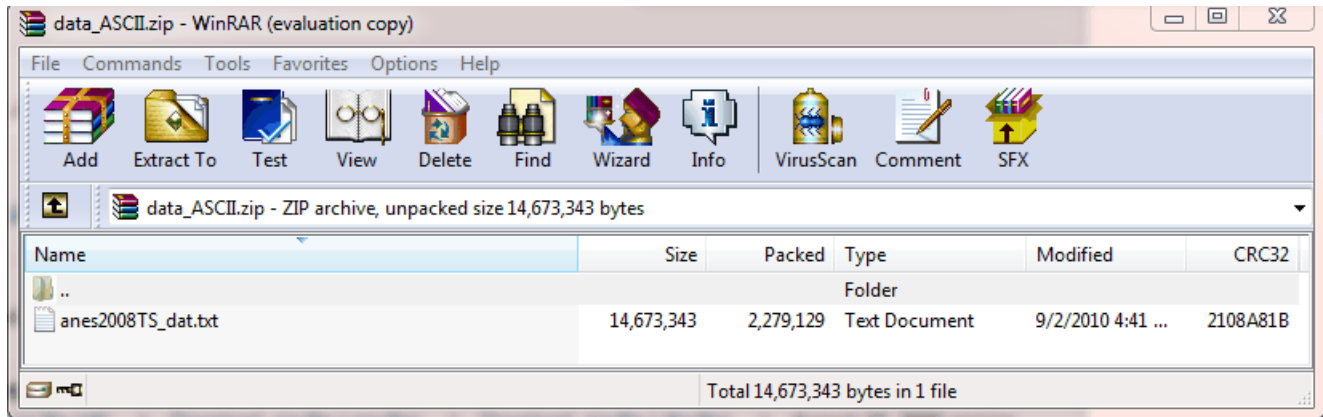


Figure 10: 2008 ANES Time Series Data .zip File

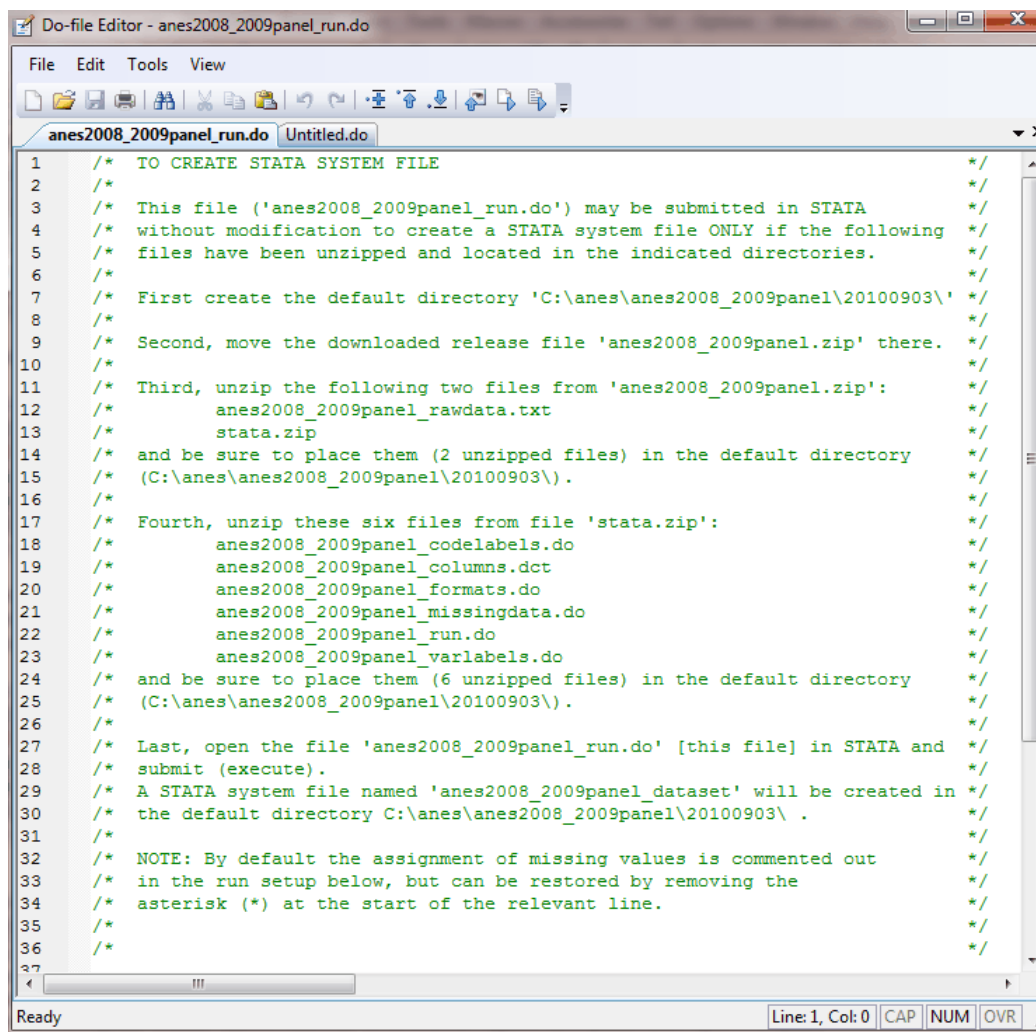


Figure 11: 2008 ANES Time Series Do-File (Run)

directory path:

```
C:\ANES\anes2008TS\20100902
```

This will require you to: (1) create a new folder on the C:\ drive called “ANES”; (2) Open the “ANES” folder and create a new folder within it called “anes2008TS”; and (3) Open the “anes2008TS” folder and create a new folder within it called “20100902”. You have now created the necessary directory path. Finally, you must place all of the necessary files within the “20100902” folder — this includes all the files that were within the *anes2008TSprepost-2.zip* file, the sub .zip file called *stata_statements.zip*, and the sub .zip file called *data_ASCII.zip*. Once you have placed all the necessary files within the directory path:

```
C:\ANES\anes2008TS\20100902
```

open the Do-File called “anes2008TS_run” and click on execute. Stata will now run a series of Do-Files and commands to use the .dct file to change the .txt file into a .dta file and save it within

```
C:\ANES\anes2008TS\20100902.
```

On a final note, the codebook for this dataset is located within a .zip file within the original .zip file called *codebook_MSword.zip*. You can choose whether you want the .txt (ASCII) version of the codebook or the Microsoft Word version of the codebook. I chose the MS Word version and unzipped the three codebook files into the

```
C:\ANES\anes2008TS\20100902
```

directory. You will need the codebook in order to understand the how each variable in the dataset is operationalized.

At this point you are free to retrieve the .dta file and place it in any directory you wish. I also recommend using Stata to create a .csv file from the .dta file so that you have a backup copy of the raw data. To do this simply click on File ⇒ Export ⇒ Comma or Tab Separated data ⇒ Leave the “Variables” box empty to make sure that all variables are chosen ⇒ Choose the file directory path and file name (make sure to use the drop down menu in the “save as” box to select a .csv file) ⇒ Click the “Comma-separated (instead of tab-separated) format” under the delimiter section ⇒ Click on “Output numeric values (not labels) of labeled variables” ⇒ Finally, click “Submit” or “OK”. Now you will have saved a .csv file which you can use as a backup. The 2008 ANES Time Series Study is now ready to be manipulated within Stata. The ANES data set is not the only data which will require this type of operation — much of the data from The Interuniversity Consortium for Political and Social Research (ICPSR) will require a similar operation. The next section will work with this data to demonstrate some basic statistics commands in Stata.

3.2 Labeling Variables

Before we move on to manipulating data it is important to understand how to label your variables in a more intuitive way. For example, in the ANES data file we just created, the variable “V081102” is the Race of the respondent. This particular ANES data file has full labels for each variable; however, there will be times when you want to label your data. The following commands can be used to label your variables and even give them value labels:

Provide A Label for Your Variables:

- | **label variable** *your variable name* “The label you would like to give your variable”
| — This will allow the user to label their variables.

Replace a Variable’s Numeric Value with a Categorical Name (Requires Both Commands):

- | **label define** *your label name* numeric value “label” numeric value “label” ...| — This

tells Stata to add a new value label. Eg. | **label define** *yesno* 0 “yes” 1 “no” |.

- | **label values** *variable1 variable2... the formerly created label you want to apply* | — This tells Stata to apply a label to a set of variables. Eg. | **label values** *variableXyesno* |.

After using the value labels set of commands you will only be able to see your label names in place of the numeric values of the variable. To browse your data without label values you can use the command | **browse** *variable1 variable2..., nolabel* |. One can also label the entire dataset with the following commands:

Provide a Label for You Entire Dataset:

- | **label data** *your data label* | — This tells Stata to give a specific label to the entire dataset.

3.3 Summary Statistics and Histograms

Summary statistics will allow the user to view several important properties of your data, such as mean, standard deviation, number of observations, minimum value, and maximum value. To view the summary statistics of any of your variables simply type | **summarize** *variable name* |. For example if you wanted to see the summary statistics associated with the race (V081102) variable you would type | **sum** V081102 |. Users may also want to tabulate different variables or cross-tabulate sets of variables. To do this you will need the command | **tabulate** *variable1 variable2...* |. For example, say we want to cross-tabulate the race and gender variables from the ANES dataset above. We would use the command | **tab** V081101 V081102 |. Stata would then output the cross-tabulation which appears in Figure 12.

This manual has its limits of course, so I am not able to mention every option of every command. However, I want to stress that the **help** command in Stata will allow users to

```
. tab v081101 v081102
```

HHList.1. Respondent: gender	HHList.2. Respondent: race						Total
	1. white	2. Black/	4. other	5. white	6. Black	7. white,	
1. Male respondent se	627	238	122	7	2	0	996
2. Female respondent	815	345	140	9	4	2	1,315
Total	1,442	583	262	16	6	2	2,311

Figure 12: Cross-Tabulation Gender and Race

view all the options associated with each command. For example, the **tab** command has some useful options. Typing `| tab variable1 variable2, column row |` will allow the user to view the relative frequency for each column and row, respectively. Viewing the help file for each command will allow you to get acquainted with each command's options, which will often come in handy.

Finally, graphing a histogram can give you a good visual representation of the distribution of your data. In Stata it is easy to generate a histogram. The command is `| hist your variable name |`. Figure 13 shows the histogram of the 7-point self-identified ideology variable in the ANES data.

3.4 Generate and Recode

This section will briefly cover how to generate and recode new and existing variables. Sometimes it will be necessary to alter the way variables are constructed. For example, in the ANES dataset the "Gender" (V081101) variable is coded as 1=male and 2=female; as you may know, dummy variables (variables that take on 2 discrete categories) often take on the values of [0,1]. If we want to use Gender as a dummy variable in a regression analysis we will want to recode it. To do this we will use the **generate** and **recode** commands. It is possible to use only the **recode** command to perform the desired operation; however, I recommend generating a new variable and keeping the old variable intact. The reason is that often times you will collapse a categorical variable with more than 2 categories into a

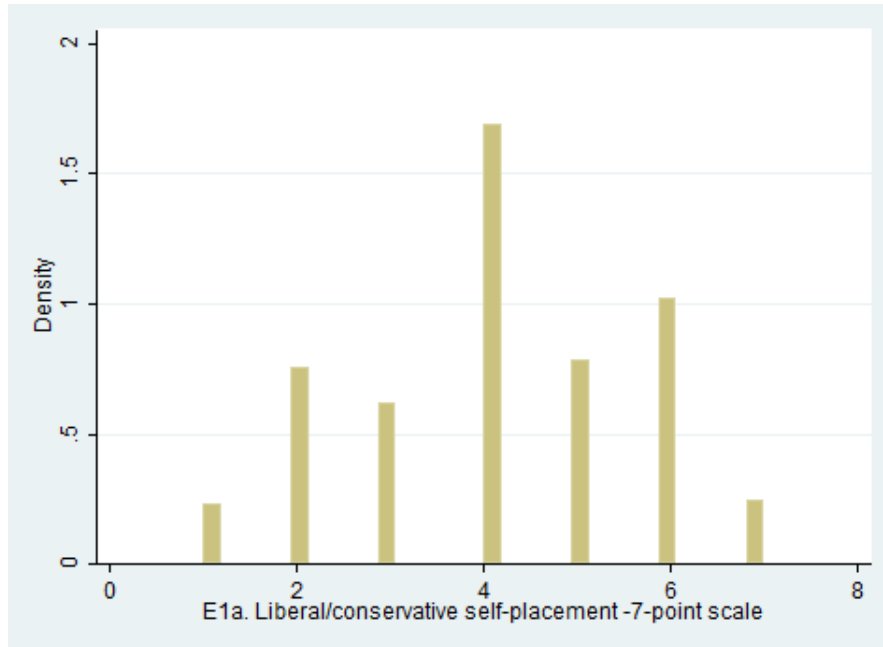


Figure 13: Histogram 7-Point Self Identified Ideology

dummy variable. If you use the recode command without generating a new variable first, you will eliminate the original variable which you may need to use in its original form in future analysis. Thus, I strongly recommend generating a new variable which is the same as the variable you want to recode and then recoding the new variable, leaving the old variable intact. Let me demonstrate what I mean. The following commands are used to transform the ANES V081101 (Gender) variable into a standard dichotomous variable.

Generate and Recode

- | **generate** gender=V081101 | — This will create a new variable named “gender” which is exactly the same as “V081101”.
- | **replace** gender=. if V081101==. | This tells Stata to replace any missing variables in the original file with missing data in the new file as well, this is a necessary command to keep the integrity of your variables.
- | **recode** gender 1=0 2=1 | — This will recode the gender variable so that male=0

and female=1, instead of male=1 and female=2. You now have a standard dummy variable to use in your analysis.

The **generate** command has the more general Stata syntax: | **generate** [*type*] *new variable name* =*exp* |, where generate creates a new variable. The values of the variable are specified by =*exp*. The **replace** command is used to change the contents of an existing variable. As seen above, I am using the replace command to make sure that all missing data in the original data is also missing in the new variable. The **recode** command is extremely useful; it is used to change the values of numeric variables according to the rules specified. Values that do not meet any of the conditions of the rules are left unchanged, unless an *otherwise* rule is specified. The general syntax for recode is: | **recode** *your variable* (rule) [(rule) ...] |.

Another example should be helpful. In the ANES data “V081102” contains data on each respondent’s race. Using the tab command we can see that: 1=White, 2=Black, 4=Other Race, 5=White and Other Race, 6=Black and other Race, and 7=White, Black and other Race. We want to make this a useful categorical variable to be used in statistical analysis. One way to do this is to create three Dummy Variables: (1) All Other=0, White=1; (2) All Other=0, Black=1; and (3) White and Black=0, All Other=1. Each set of code needed to make each variable is shown below:

White:

- | **gen** White=V081102 |
- | **replace** White=. if V081102==. |
- | **recode** White 2=0 4=0 5=0 6=0 7=0 1=1 |

Black:

- | **gen** Black=V081102 |

- | **replace** Black=. if V081102==. |
- | **recode** Black 1=0 2=1 4=0 5=0 6=0 7=0 |

All Other:

- | **gen** Other_Race=V081102 |
- | **replace** Other_Race=. if V081102==. |
- | **recode** Other_Race 1=0 2=0 4=1 5=1 6=1 7=1 |

Now we have a set of Dummy Variables which can be used in a regression analysis. Recall that the interpretation of any of these variables is against the out-group. Therefore, if we use the variable *White* the coefficient would be interpreted in comparison to both African Americans and all other races in the original variable. Learning how to generate and recode existing variables is vital to using Stata for data manipulation.

4 Hypothesis Testing

Stata can easily conduct one-sample and two-sample hypothesis tests. We can test a mean and test differences in means. A one-sample or single-sample hypothesis test, tests a claim surrounding a variable's mean. The Stata syntax for a single-sample hypothesis test is | **ttest** *varname* |. The single-sample t-test compares the mean of the sample to a given number (which you supply). The independent samples t-test compares the difference in the means from the two groups to a given value (usually 0). The dependent-sample or paired t-test compares the difference in the means from the two variables measured on the same set of observations to a given number (usually 0), while taking into account the fact that the scores are not independent.

For example, let's say we think that the population mean for for the variable "percent of a state that voted for the democratic presidential candidate" (pervotedem) in the

dataset I compiled from the U.S. Statistical Abstract is equal to 50 percent. To test this hypothesis in Stata we would type | `ttest pervotedem==50` |. The output/results can be viewed below.

```
. ttest pervotedem==50
One-sample t test
```

variable	obs	Mean	Std. Err.	Std. Dev.	[95% Conf. Interval]	
pervot~m	153	47.91804	.8659288	10.71095	46.20723	49.62885

```

      mean = mean(pervotedem)
Ho: mean = 50
      t = -2.4043
      degrees of freedom = 152

      Ha: mean < 50
Pr(T < t) = 0.0087

      Ha: mean != 50
Pr(|T| > |t|) = 0.0174

      Ha: mean > 50
Pr(T > t) = 0.9913

```

Stata calculates the t-statistic and its p-value under the assumption that the sample comes from an approximately normal distribution. If the p-value associated with the t-test is small (0.05 is often used as the threshold), there is evidence that the mean is different from the hypothesized value. If the p-value associated with the t-test is not small ($p > 0.05$), then the null hypothesis is not rejected and you can conclude that the mean is not different from the hypothesized value (UCLA, N.d.) (The UCLA Academic Technology Services “Statistical Computing” resources linked to this citation is great for a variety information on Stata and statistics). Stata automatically runs 3 tests. Note that “!” is “*not equal*” in Stata language. In the output/results, **Ho** is the null hypothesis that is being tested. The single sample t-test evaluates the null hypothesis that the population mean is equal to the given number. Each **Ha** is an alternative hypothesis which Stata tests for. In this example, we reject the null that the population mean for our variable is equal to 50 percent and accept the alternative hypothesis that the mean *is below* 50 percent. We can also accept the alternative hypothesis that the mean is *not equal* to 50 percent. Finally, we must accept the null hypothesis and reject the alternative hypothesis that the population mean is greater than 50 percent. The output/results gives all the information needed to

varname, by(*varname*) |. For example, below I am going to compare the percent of the U.S. states' population that voted for the Democratic candidate between Southern and non-Southern states. The test assumes that variances for the two populations are the same. The interpretation for p-value is the same as in the other types of t-tests (and the observations are supposed to be randomly selected from the larger population of observations — with observational data we run into the problem of *non-random assignment*. See Arena (2009) for a summary).

```
. ttest pervotedem, by(south)
```

```
Two-sample t test with equal variances
```

Group	Obs	Mean	Std. Err.	Std. Dev.	[95% Conf. Interval]	
0	108	49.65741	1.138354	11.83012	47.40075	51.91406
1	45	43.74356	.8264415	5.543938	42.07797	45.40914
combined	153	47.91804	.8659288	10.71095	46.20723	49.62885
diff		5.913852	1.844993		2.268516	9.559187

```
diff = mean(0) - mean(1)
Ho: diff = 0
Ha: diff < 0
Pr(T < t) = 0.9992

Ha: diff != 0
Pr(|T| > |t|) = 0.0016

t = 3.2054
degrees of freedom = 151

Ha: diff > 0
Pr(T > t) = 0.0008
```

The above output/results indicates that the we must accept the null hypothesis for the first t-test. The difference in means for the two groups (Southern and non-Souther states) is not less than zero. For the second t-test we can reject the null and accept the alternative hypothesis that the difference in mean between the two groups is not equal to zero. Finally, we can reject the null and accept the hypothesis for the third t-test that the mean difference between the two groups is greater than zero. Non-Southern states between the years of 2006 and 2008 had a larger mean for percent of the population that voted democrat than for Southern states. Thus, non-Southern states may be less likely to have a larger percent of the their population vote for the Democratic party's candidate.

Another primary tool for most researchers is Ordinary Least Squares (OLS) Regression. To perform an OLS Regression in Stata you type: `| reg depvar, indepvar1 indepvar2 indepvar3..., options |`. One useful option when using the `reg` command is **beta**; using the **beta** option tells Stata to report the standardized coefficients as well as unstandardized coefficients. Standardized coefficients are the estimates resulting from an analysis carried out on variables that have been standardized so that their variance is 1. This means that they are in “standard deviation” terms or units and can be compared to each other. Recall that unstandardized coefficients literally tell us the change in Y for every 1 unit change in X. The problem is that more often than not the units of analysis for all of your variables are different (for example dollars and percentage unemployed) and they cannot be directly compared to each other. When the coefficients are standardized in standard deviation terms they can be directly compared to each other and the researcher can tell which of the variables has the largest impact. Figure 14 shows an example OLS regression analysis which was performed using a set of recoded variables from the ANES 2008 time series study. The Dependent Variable is the Felling Thermometer Score (Which ranges from 0-100, so it is continuous) for President Obama. Table 1 uses the command `| outtex, detail legend |` to produce the \LaTeX code which can be copied and pasted directly into a .TeX document. To use this command you will need the `outtex` plug-in which can be found by typing `| findit outtex |` and then installing the package as discussed previously. Figure 15 shows the \LaTeX “**outtex**” code output which can be copied and pasted into any .TeX file.

5.1 Simple Post Estimation Commands

After you have run your regression you may want to test for some common modeling problems. One such problem is multi-collinearity; this occurs when multiple explanatory variables are moderate-highly correlated with each other. When this occurs the variables provide redundant information. The consequences include inflated standard errors for

```
. reg feelobam Black gender news appecon appfor betwor lib dem edu chrchatt, beta
```

Source	SS	df	MS		Number of obs = 455
Model	188240.312	10	18824.0312		F(10, 444) = 45.11
Residual	185295.917	444	417.333146		Prob > F = 0.0000
					R-squared = 0.5039
					Adj R-squared = 0.4928
Total	373536.229	454	822.767023		Root MSE = 20.429

feelobam	Coef.	Std. Err.	t	P> t	Beta
Black	15.52699	2.410054	6.44	0.000	.2381542
gender	7.72068	1.955983	3.95	0.000	.1337422
news	2.87976	2.396865	1.20	0.230	.0406913
appecon	-1.771885	2.914245	-0.61	0.543	-.0245301
appfor	-17.22937	2.743603	-6.28	0.000	-.270381
betwor	-1.182389	1.089743	-1.09	0.279	-.0370435
lib	5.749615	2.459268	2.34	0.020	.0874572
dem	16.72009	2.529779	6.61	0.000	.2891408
edu	-.4931043	2.219957	-0.22	0.824	-.0076238
chrchatt	-5.220991	1.980886	-2.64	0.009	-.091109
_cons	57.08152	3.380439	16.89	0.000	.

Figure 14: Sample OLS Regression

```
. outtex, details legend
%----- Begin LaTeX code -----%
{
\def\onepc{${\ast\ast}$} \def\fivepc{${\ast}$}
\def\tenpc{${\dag}$}
\def\legend{\multicolumn{4}{l}{\footnotesize{significance levels
:\hspace{1em} ${\dag}$ : 10% \hspace{1em}
${\ast}$ : 5% \hspace{1em} ${\ast\ast}$ : 1% \normalsize}}
\begin{table}[htbp]\centering
\caption{Estimation results : regress}
\label{tabresult regress}
\begin{tabular}{l r @{} l c } \hline \hline
\multicolumn{1}{c}
{\textbf{variable}}
& \multicolumn{2}{c}{\textbf{Coefficient}} & \textbf{(Std. Err.)} \\ \hline
female & 7.483\onepc & (2.045)\
news & 1.953& & (2.506)\
appecon & -1.533& & (3.039)\
appfor & -19.804&\onepc & (2.812)\
betwor & -1.152& & (1.139)\
lib & 4.751&\tenpc & (2.559)\
dem & 21.078&\onepc & (2.544)\
edu & -1.473& & (2.315)\
chrchatt & -4.757&\fivepc & (2.072)\
Intercept & 60.632&\onepc & (3.467)\
\hline
\multicolumn{4}{c}{\hspace{1em}}
\hline N & \multicolumn{3}{c}{457}\
R^2 & \multicolumn{3}{c}{0.452}\
F_{(9,447)} & \multicolumn{3}{c}{41.025}\
\hline
\legend
\end{tabular}
\end{table}
}
%----- End LaTeX code -----%
```

Figure 15: L^AT_EX Code

Table 1: OLS Regression: President Obama Feeling Thermometer Score

Variable	Coefficient	(Std. Err.)
African American	15.527**	(2.410)
Female	7.721**	(1.956)
Read News Paper > 4 Times Per Week	2.880	(2.397)
Approve of Economy	-1.772	(2.914)
Approve of Foreign Policy	-17.229**	(2.744)
The Economy is Better in Last Year	-1.182	(1.090)
Liberal	5.750*	(2.459)
Democrat	16.720**	(2.530)
Education \geq BA	-0.493	(2.220)
Attend Church More than 3 Times a Month	-5.221**	(1.981)
Intercept	57.082**	(3.380)
<hr/>		
N	455	
R ²	0.504	
F (10,444)	45.106	

Significance levels: † : 10% * : 5% ** : 1%

the unstandardized coefficients, incorrect p-values, and sometimes sign switches. One way to test for this problem is to run a correlation matrix of all your independent variables using the `| corr var1 var2 var3... |`. This will allow you to look over the correlations between your independent variables and see whether some of them are moderate-highly correlated (a reasonably high correlation can be thought of as around .40 and higher). Another common method is the computation of Variance Inflation Factor scores (VIFs). To compute VIFs in Stata use the command `| estat vif |` after running your regression. The general rule is that individual or average VIF scores greater than 10 indicate multicollinearity; although VIF scores over 2 have been known to be problematic.

Another important problem to test for is heteroscedasticity; OLS models make the assumption that the variance in the error term is constant (homoscedastic). Heteroscedasticity (also written heteroskedasticity) violates this assumption of OLS and occurs when errors increase as the value of an independent variable increases. It is vital to test for violations of the Gauss-Markov assumptions, one of which is the absence of heteroscedastic-

ity; thus making sure that we have the best linear unbiased estimate (BLUE). In Stata we use the command | **estat hettest** | after running the regression. This will run a Breusch-Pagan/Cook-Weisberg test for heteroskedasticity. The null hypothesis for this test is constant variance. Thus, if the test comes out statistically significant — $\text{prob} > \chi = 0.005$ or less — then the alternate hypothesis must be accepted, which would indicate non-constant variance and heteroskedasticity. When heteroskedasticity is a problem you can use robust standard errors to correct for the issue. To do this in Stata use the “r” option after your standard regression command: | **reg depvar indvar1 indvar2 indvar3, r** |. Placing the “r” in the options syntax will tell Stata to use robust standard errors when estimating the model.

6 Advanced Regression Models

There will be many times when you need to add additional specifications to OLS (for example robust standard errors above). For example, we want our independent variables to explain as much about an observation as possible; however, sometimes there is some unmodeled effect which goes into the error term. In some circumstances this un-modeled effect may have to do with properties within our observations; for example, if the unit of analysis is U.S. States, something about Texas or New York (which is laden and unmodeled) may have a systematic effect on our prediction. To account for this we can run a *fixed effects* regression model using the | **xtreg** | command. The syntax for a fixed effects model is | **xtreg depvar indvar1 indvar2 indvar3, fe** |. This is essentially the same as adding a dummy variable in the model for each case (i.e. U.S. States in my example). The output/results are interpreted in the same way, except Stata gives us another F-test which indicates whether the variance explained by each “group” is statistically significant in the aggregate. Moreover, when using the | **xtreg depvar indvar1 indvar2 indvar3, fe** | command Stata presents an F-statistic only for those terms in which we are interested (the

ones that are listed in the output) instead of showing the F-test including the absorbed terms (i.e. dummies for each observation). You can also use a *random effects* model which essentially thinks of each intercept as the result of a random deviation from some mean intercept. The command syntax for this model is `| xtreg depvar indvar1 indvar2 indvar3, re |`. However, this method requires that we treat the u_i (unobserved individual-specific effect) terms as random variables and that they follow the normal distribution — this is not always the case. Both of these models are controlling for unobserved individual or observation-specific effects and are commonly used.

When using time-series cross-sectional data (i.e. panel-data), when we have the same observations over multiple time periods (days, weeks, months, years) you should use panel-corrected standard errors as advocated by Beck and Katz (1995). The Stata command for this regression is `| xtpcse depvar indvar1 indvar2 indvar3 |`. Moreover, in time-series models you may have serial-autocorrelation. Autocorrelation means correlation between successive values in the data (past data points correlate with the subsequent data point). It mainly occurs when data is measured over time, and the values are not independent of each other. To correct for this you can use the command `| xtpcse depvar indvar1 indvar2 indvar3, correlation (ar1) |`. This is a common model used to estimate a time-series OLS with panel corrected standard errors and autocorrelation correction.

7 Limited Dependent Variables Regression Models

In this section I will discuss two models — Logit and Probit — that are used when the dependent variable is categorical or limited in the sense that it takes on a finite number of discrete values, which can represent nominal categories. An example of a limited dependent variable would be *home ownership*; one either owns a home or they do not. Another example commonly used in political science is vote choice (i.e. Democrat or Republican

Candidate) or whether an individual voted at all (i.e. Voted or \sim Voted). For International Relations students a common example would be whether a war or crisis occurred or not. It is clear that these models, which are sometimes called qualitative response models, are important for political science research.

When using an OLS Regression model, we know that the Population Regression Function (PRF) is:

$$Y_i = \beta_1 + \beta_2 X_i + \beta_3 X_i \dots \beta_n X_i + u_i \quad (1)$$

We also know that the Sample Regression Function (SRF) is:

$$\hat{Y}_i = \hat{\beta}_1 + \hat{\beta}_2 X_i + \hat{\beta}_3 X_i \dots \hat{\beta}_n X_i + \hat{u}_i \quad (2)$$

In these models the coefficients can be directly interpreted as “a one unit change in X_i , produces a coefficient change in Y ”. In this section we explore models where the coefficients cannot be interpreted directly; instead they require a transformation based on an underlying distribution function. First, we will look at the logistic distribution function and then we will look at the probit model and the cumulative normal distribution.

7.1 Logistic Regression

The logistic distribution function is:

$$P_i = E(Y = 1|X_i) = \frac{1}{1 + e^{-(\beta_1 + \beta_2 X_i)}} \quad (3)$$

This can be written:

$$P_i = \frac{1}{1 + e^{-(\beta_1 + \beta_2 X_i)}} = \frac{e^{(\beta_1 + \beta_2 X_i)}}{1 + e^{(\beta_1 + \beta_2 X_i)}} \quad (4)$$

This could be the probability of voting Republican based on the independent variables

in the model. Thus, $(1 - P)$ would be the probability of not voting Republican, which can be written:

$$1 - P_i = \frac{1}{1 + e^{(\beta_1 + \beta_2 X_i)}} \quad (5)$$

Finally,

$$\frac{P_i}{1 - P_i} = \frac{1 + e^{(\beta_1 + \beta_2 X_i)}}{1 + e^{-(\beta_1 + \beta_2 X_i)}} = e^{(\beta_1 + \beta_2 X_i)} \quad (6)$$

is the odds ratio in favor of voting Republican, keeping with the example from above. If we take the natural of equation 6 we get:

$$L_i = \ln\left(\frac{P_i}{1 - P_i}\right) = \beta_1 + \beta_2 X_i \quad (7)$$

These equations and an in-depth explanation of the logistic model can be found in Gujarati (2003), which is where this discussion was drawn. L , which is the log odds ratio, is linear in X and also linear in the parameters. L_i is the logit model, which is commonly used to estimate modes with limited dependent variables. The next section will cover a logit example, predicting a Respondent's Vote for the Republican Party in the 2008 Presidential election.

7.2 Logit Example

The logit command is similar to the regress command for OLS. The Stata syntax for a logit regression is | **logit** depvar indvar1 indvar2 indvar3..., options |. Table 2 is the logistic regression output for a model that predicts a Respondent's probability of voting for the Republican Candidate in 2008 (John McCain). In this case the dependent variable equaled 1 if the respondent voted for John McCain and 0 otherwise.

The direction of the relationship can easily be obtained by looking at the sign of the co-

Table 2: Logit Model, Respondent Vote Republican in 2008

Variable	Coefficient	(Std. Err.)
Republican = 1 (Includes Leaners)	2.711**	(0.217)
African American = 1	-3.738**	(0.740)
Approve of Pres. Handle Economy = 1	1.475**	(0.273)
Liberal to Conservative 1-7	0.487**	(0.080)
Intercept	-3.685**	(0.338)
<hr/>		
N	1116	
Log-likelihood	-331.639	
$\chi^2_{(4)}$	816.859	
<hr/>		
Significance levels : † : 10% * : 5% ** : 1%		

efficient. For example, there is a negative association between a respondent being African American and voting for John McCain. There is a positive association between a respondent saying that they approve of the way the President (in this case George W. Bush) handled the economy and voting for John McCain. However, we want to know more than the association, we want to know the probability that a respondent will vote for John McCain given that they approved of the way George W. Bush handled the economy or the probability of voting Republican based on different parameters of the other independent variables. This requires the logistic distribution function or P_i above from equation 4. e is the Base of Natural Logarithms and is approximately 2.71828. Let's say that we wanted to calculate the probability that a respondent would vote Republican based on the logistic regression model estimated in Table 2. I will estimate the probability for the following values of the independent variables:

- Republican = 1.
- African American = 0.
- Approve of the Economy = 1.
- Liberal to Conservative = 5 (Slightly Conservative).

To do this we need to utilize equation 4:

$$P_{VoteRep} = \frac{2.71828^{(-3.685+2.711*1+-3.738*0+1.475*1+0.487*5)}}{1 + 2.71828^{(-3.685+2.711*1+-3.738*0+1.475*1+0.487*5)}} = 0.949597528 \quad (8)$$

In this case a respondent with the above characteristics based on the set of independent variables has about a 95% chance of voting for the Republican Candidate (John McCain). If we keep everything the same, except change the Liberal-Conservative value from 5 (slightly conservative) to 7 (very conservative), the same respondent would have about a 98% chance of voting for John McCain. If we change the value of African American to 1 (indicating that the respondent was African American) and keeping the Liberal-Conservative value at 7 (indicating a very conservative respondent) the predicted probability of voting Republican drops to 0.542 or approximately 54.2%. If the respondent was African American and only slightly conservative (5 on the liberal-conservative scale), the predicted probability drops to about 33%. Using equation 4, we can manually calculate the predicted probabilities for certain values of our independent variables. However, as stated before, Tomz, Wittenberg and King (2003) have developed the *Clarify* add-on which assists with calculating predicted probabilities using three simple commands.

The *Clarify* commands are as follows:

- | **estsimp** | (estimates the model and simulates its parameters).
- | **setx** | (sets values of Xs before simulating quantities of interest).
- | **simqi** | (simulates quantities of interest).

For example, if we wanted to estimate the same quantity as equation 8 we would use the following set of commands:

- | **estsimp logit** VoteRep Rep Black AppPresEcon LibCon |.
- | **setx** Rep 1 Black 0 AppPresEcon 1 LibCon 5 |.
- | **simqi** |.

Quantity of Interest	Mean	Std. Err.	[95% Conf. Interval]	
Pr (VoteRep=0)	.0515808	.0136643	.0286384	.0815726
Pr (VoteRep=1)	.9484192	.0136643	.9184274	.9713616

Figure 16: Logit Predicted Probabilities with Clarify

Measures of Fit for logit of VoteRep			
Log-Lik Intercept only:	-740.069	Log-Lik Full Model:	-331.639
D(1111):	663.279	LR(4):	816.859
		Prob > LR:	0.000
MCFadden's R2:	0.552	MCFadden's Adj R2:	0.545
Maximum Likelihood R2:	0.519	Cragg & Uhler's R2:	0.707
McKelvey and Zavoina's R2:	0.732	Efron's R2:	0.631
Variance of y*:	12.293	Variance of error:	3.290
Count R2:	0.892	Adj Count R2:	0.713
AIC:	0.603	AIC*n:	673.279
BIC:	-7133.170	BIC':	-788.789

Figure 17: SPost Model Fit Commands (Logit)

The Stata output/results appear in Figure 16 and show the predicted probability of voting Republican based on the values of the independent variables stated above. Notice that *Clarify* also calculates the mean, standard error, and confidence intervals around the prediction. *Clarify* will also calculate the predicted probabilities for each respondent. This can also be done manually with the `| predict |` command discussed above.

Another great set of add-on commands for limited dependent variables is the *SPost* package, from Long (1997). As we know, the pseudo R^2 statistic that Stata reports is not the best measure of model fit for logit and probit models. There are several others, for example the Count R^2 is $\frac{\text{Number of Correct Predictions}}{\text{Total Number of Observations}}$. The Count R^2 and other measures of fit can be computed easily with the *SPost* package. After finding and installing the *SPost* package, run the `| fitstat |` command to get the output/results in Figure 17.

Quantity of Interest	Mean	Std. Err.	[95% Conf. Interval]	
Pr (VoteRep=0)	.0544235	.0153344	.0275821	.0882126
Pr (VoteRep=1)	.9455765	.0153344	.9117874	.9724179

Figure 18: Probit Predicted Probabilities with Clarify

7.3 Probit Example

The probit model uses the normal cumulative distribution function (CDF) to estimate models with limited dependent variables. Probit predicted probabilities are more complicated to calculate by hand than their logit counterpart. Therefore, instead of focusing on the math behind the Probit analysis I will simply demonstrate the commands necessary to run a Probit model, which will be identical to the Logit model above. The Stata syntax for a Probit model is `| probit depvar indvar1 indvar2 indvar3..., options |`.

Table 3: Probit Model, Respondent Vote Republican in 2008

Variable	Coefficient	(Std. Err.)
Republican = 1 (Includes Leaners)	1.607**	(0.123)
African American = 1	-1.648**	(0.281)
Approve of Pres. Handle Economy = 1	0.760**	(0.145)
Liberal to Conservative 1-7	0.256**	(0.043)
Intercept	-2.033**	(0.175)
<hr/>		
N	1116	
Log-likelihood	-335.177	
$\chi^2_{(4)}$	809.785	
<hr/>		
Significance levels: † : 10% * : 5% ** : 1%		

The model in Table 3 is the probit estimation of the model used in the Logit example. Figure 18 represents the predicted probabilities using *Clarify* for the example used above. Figure 19 represents the model fit statistics using *SPost*. Notice that the results are similar but not identical. Moreover, the commands to conduct the analysis are the same as in the logit example; one just inserts “probit” for the former “logit” command. There are also multinomial and ordered logit and probit models, which will be discussed next.

Log-Lik Intercept only:	-740.069	Log-Lik Full Model:	-335.177
D(1111):	670.353	LR(4):	809.785
		Prob > LR:	0.000
McFadden's R2:	0.547	McFadden's Adj R2:	0.540
Maximum Likelihood R2:	0.516	Cragg & Uhler's R2:	0.702
McKelvey and Zavoina's R2:	0.709	Efron's R2:	0.628
Variance of y*:	3.435	Variance of error:	1.000
Count R2:	0.892	Adj Count R2:	0.713
AIC:	0.610	AIC*n:	680.353
BIC:	-7126.096	BIC':	-781.715

Figure 19: SPost Model Fit Commands (Probit)

8 Graphing in Stata: A Simple Example

This section will demonstrate how to graph predicted effects in Stata and place confidence intervals around the prediction. Stata has several graphing commands that you should become familiar with. First, we can make a simple scatter plot between two variables using the the command | **graph twoway** (scatter *var1 var2*) |. This will produce a result like that in Figure 16. The other graph command that will be useful, and is demonstrated in this example, is | **graph twoway** (lfit *var1 var2*) |. This command produces a line which had been linearly fit to the data. The following example will use a regression model predicting divorce rates in the U.S. from the 2006 to 2008. The data was obtained from the U.S. Statistical Abstract.

First, we must run our regression model. Figure 17 shows the model as Stata output/results and Table 2 is a presentation quality table which shows the label for each variable. Next, we need the set of | **predict** | commands in Stata to get predicted values for each observation. Below is a set of important predict commands and a summary of their functions.

- | **predict** *varname1*, xb | Calculates a predicted values (\hat{y}) for each observation, and saves it to a new variable called "varname1".
- | **predict** *varname2*, r | Calculates the residual for each observation and saves it to a

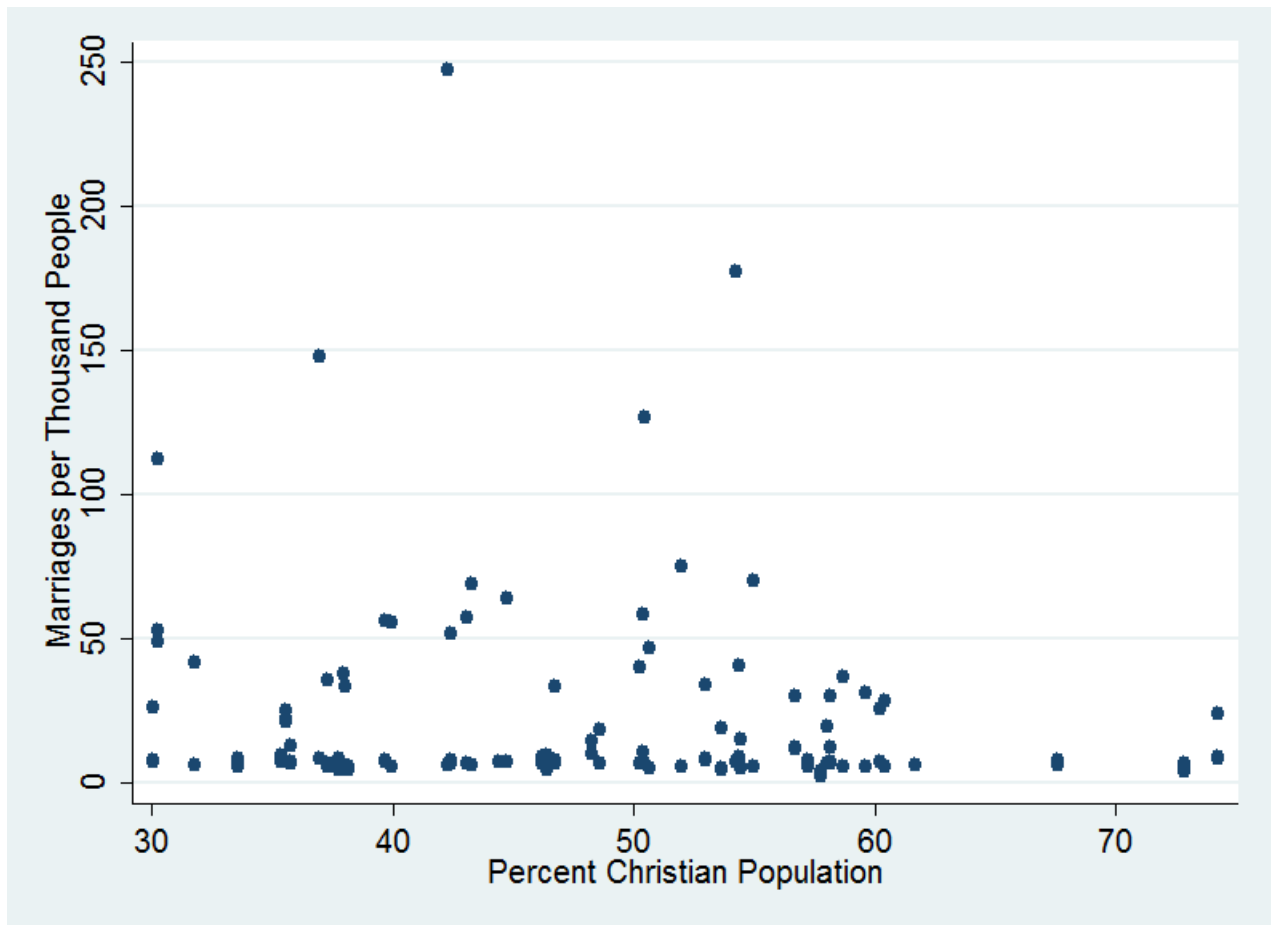


Figure 20: Scatter Plot

```
. reg numdiv1000 perunemp perchrist perjewish scpi perba illdrug age1824 black
```

Source	SS	df	MS			
Model	10509.567	8	1313.69588	Number of obs =	132	
Residual	10057.7791	123	81.7705618	F(8, 123) =	16.07	
Total	20567.3461	131	157.002642	Prob > F =	0.0000	
				R-squared =	0.5110	
				Adj R-squared =	0.4792	
				Root MSE =	9.0427	

numdiv1000	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
perunemp	-3.458638	.7300542	-4.74	0.000	-4.903736	-2.013541
perchrist	-.1284656	.0823078	-1.56	0.121	-.2913889	.0344577
perjewish	-.8132293	.7113201	-1.14	0.255	-.5947853	2.221244
scpi	-.0312924	.0084547	-3.70	0.000	-.048028	-.0145568
perba	-.1820769	.2223709	-0.82	0.414	-.6222466	.2580927
illdrug	1.091597	.5987039	1.82	0.071	-.0935005	2.276695
age1824	-.0111665	.0018827	5.93	0.000	-.0074397	-.0148933
black	-.1036816	.0801262	-1.29	0.198	-.0549233	-.2622865
_cons	26.0546	7.779372	3.35	0.001	10.65581	41.45339

Figure 21: OLS Predicting Divorce Rates

Table 4: OLS Predicting Divorce Rates

Variable	Coefficient	(Std. Err.)
Unemployment Percentage	-3.459**	(0.730)
Percent Christian Population	-0.128	(0.082)
Percent Jewish Population	0.813	(0.711)
SCPI	-0.031**	(0.008)
Percent of Population with a BA or Greater	-0.182	(0.222)
Number of Illicit Drug Crimes in Thousands	1.092 [†]	(0.599)
Percent Those Between Ages 18-24	0.011**	(0.002)
Percent African American Population	0.104	(0.080)
Intercept	26.055**	(7.779)

N	132
R ²	0.511
F _(8,123)	16.066

Significance levels: † : 10% * : 5% ** : 1%

new variable called “varname2”.

- | **predict** *varname3*, *stdp* | Calculates the standard deviation of the prediction for each observation, and saves it to a new variable called “varname3”.

We are going to use these commands to create a graph which visually illustrates the effect of the percentage of the population which are Christians on Divorce rates for the U.S. states from 2006-2008. First, I use the command | **predict** *divhat* |, which calculates a predicted value for each observation in the dependent variable. I named it “divhat” because it is a prediction for divorce rates. Now let’s take a look at the linear fit overlaying a scatter plot of our data by using the command: | **graph twoway** (*scatter divhat perchrist*) (*lfit divhat perchrist*) |; where *divhat* is our predicted values, *perchrist* is the independent variable for the percent of each state that is Christian, *scatter* is the command used to tell Stata to create a scatter plot, and *lfit* is the command used to tell Stata to plot a linearly fitted line. Figure 18 shows what this looks like.

Now we have a nice visual representation of the effect, but we are not finished yet.

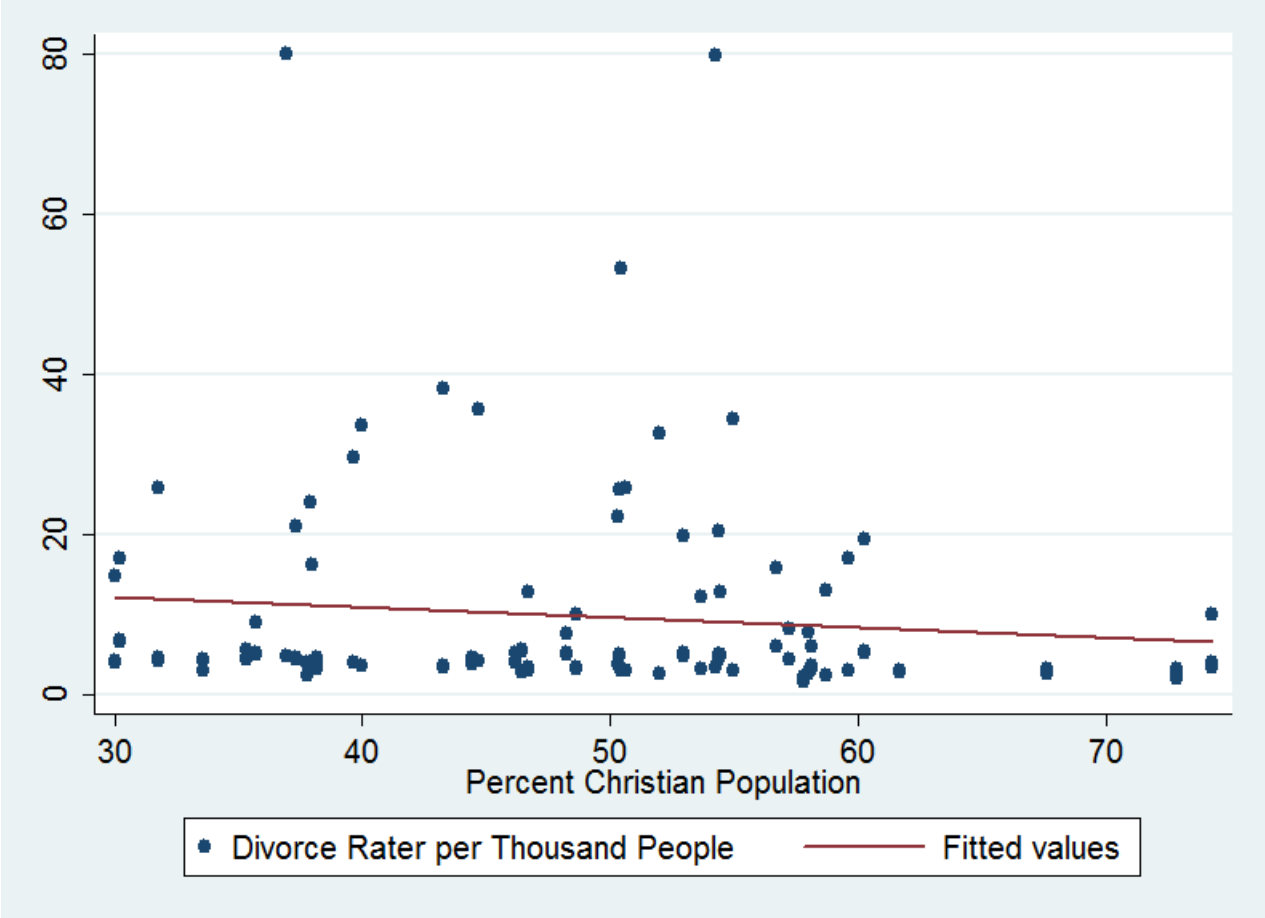


Figure 22: Scatter Plot With Fitted Overlay

Presentation is important and scholars like to see 95% confidence intervals around the estimate. To do this we will need to predict the standard deviation around each observation's prediction. Using the commands from above we type: | **predict** stdpred, stdp |, which creates the standard deviation for each predicted value and saves it as "stdpred". To create a 95% confidence interval we can simply add and subtract the product of the critical value (1.96 or the more accurate critical value reported in the regression) multiplied by our newly calculated stdppred variable from our divhat prediction. Thus, the Stata command would be:

- | **gen** lowerpred = divhat - 1.96 * stdpred | This generates a new variable for the lower-bound on our prediction which I have called "lowerpred".
- | **gen** upperpred = divhat + 1.96 * stdpred | This generates a new variable for the upper-bound on our prediction which I have called "upperpred".

Finally we can graph the effect of the predicted values with confidence intervals. The Stata programming code would be:

```
| graph twoway (scatter numdiv1000 perchrist) (lfit divhat perchrist) (lfit lowerpred perchrist) (lfit upperpred perchrist) |,
```

where each set of parentheses tells Stata to graph a separate fitted line. We will have three lines on our graph: (1) Our linearly fitted prediction, (2) Our upper bound on the prediction, and (3) Our lower bound on the prediction. Moreover, Stata will also overlay these lines on top of a scatter plot. Notice how each part of the graph appears in a separate argument governed by a set of parentheses. Lastly, we can label our X and Y axes using the same Stata code with added arguments:

```
| graph twoway (scatter numdiv1000 perchrist, msize(.2) jitter(2) ytitle(Divorce Rate per
```

Thousand People) xtitle(Percent Christian Population)) (lfit divhat perchrist) (lfit lowerpred perchrist) (lfit upperpred perchrist) |

Notice that not we have added the commands “msize(.2)”, “ytitle()”, “xtitle()”, and “jitter(2)” to the first set of parentheses. The “msize(.2)” simply tells Stata to make each point on the scatter plot smaller because the default is quite large and can make the graph look messy. The “ytitle()” and ‘xtitle()” options allow us to label the Y and X axes, respectively. Finally, the “jitter(2)” command tells Stata to place each point of the scatter plot closer to its estimate which can be used for ascetic reasons. Figure 19 represents the end result. Lastly, you can also use the | mlabel(var1) | option within the set of scatter plot parentheses to label each point on the scatter plot with its value. Figure 21 illustrates this feature; notice that it can get quite messy.

Now, as you have postulated, Stata has a “canned” package for producing graphs with confidence intervals. The command is | **graph twoway** (scatter var1 var2) (lfitci var1 var2) |. Simply adding “ci” to our *lfit* command tells Stata to calculate confidence intervals and place them around our linearly fitted estimate. Figure 20 illustrates the “canned” graph. Notice that there is a slight difference in my confidence intervals and the ones Stata calculated; this is due to the fact that I used 1.96 as the critical value (t-Statistic) instead of the precise critical value calculated by Stata. Essentially, it is up to the user to decide if they want to manually create graphs or use the canned Stata package.

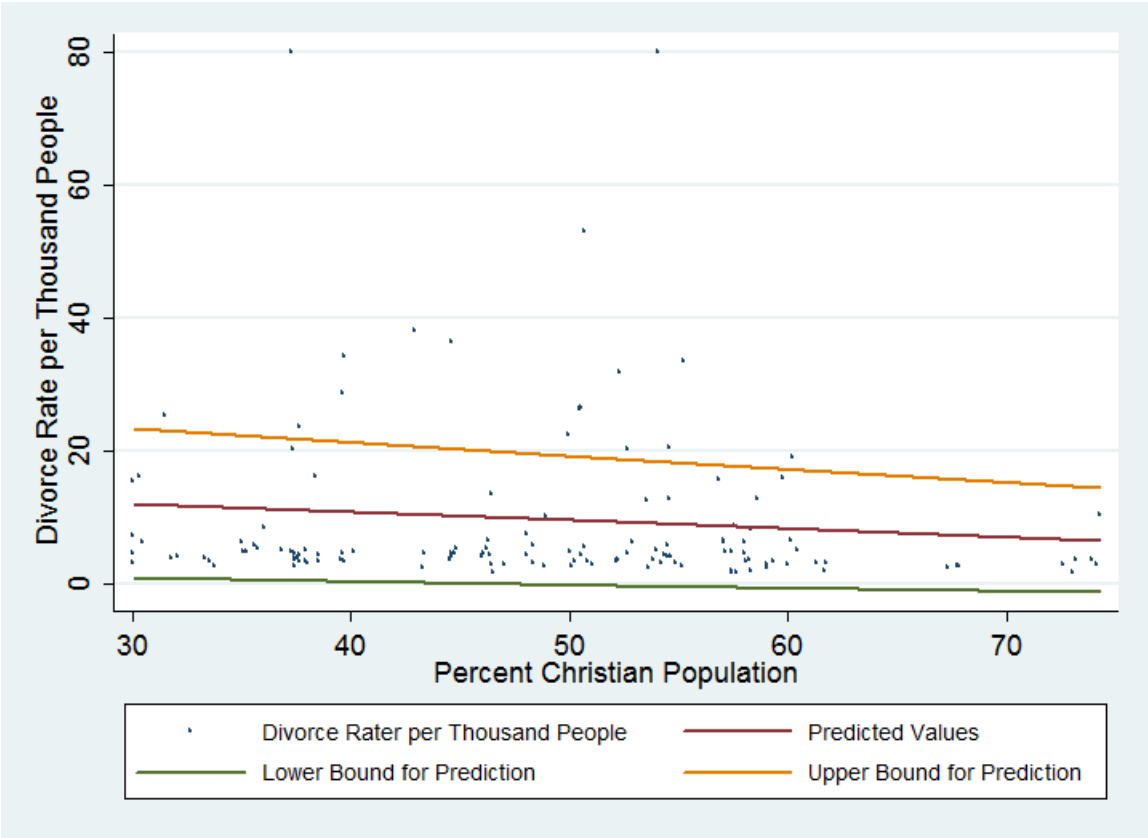


Figure 23: Estimated Effect with Confidence Intervals

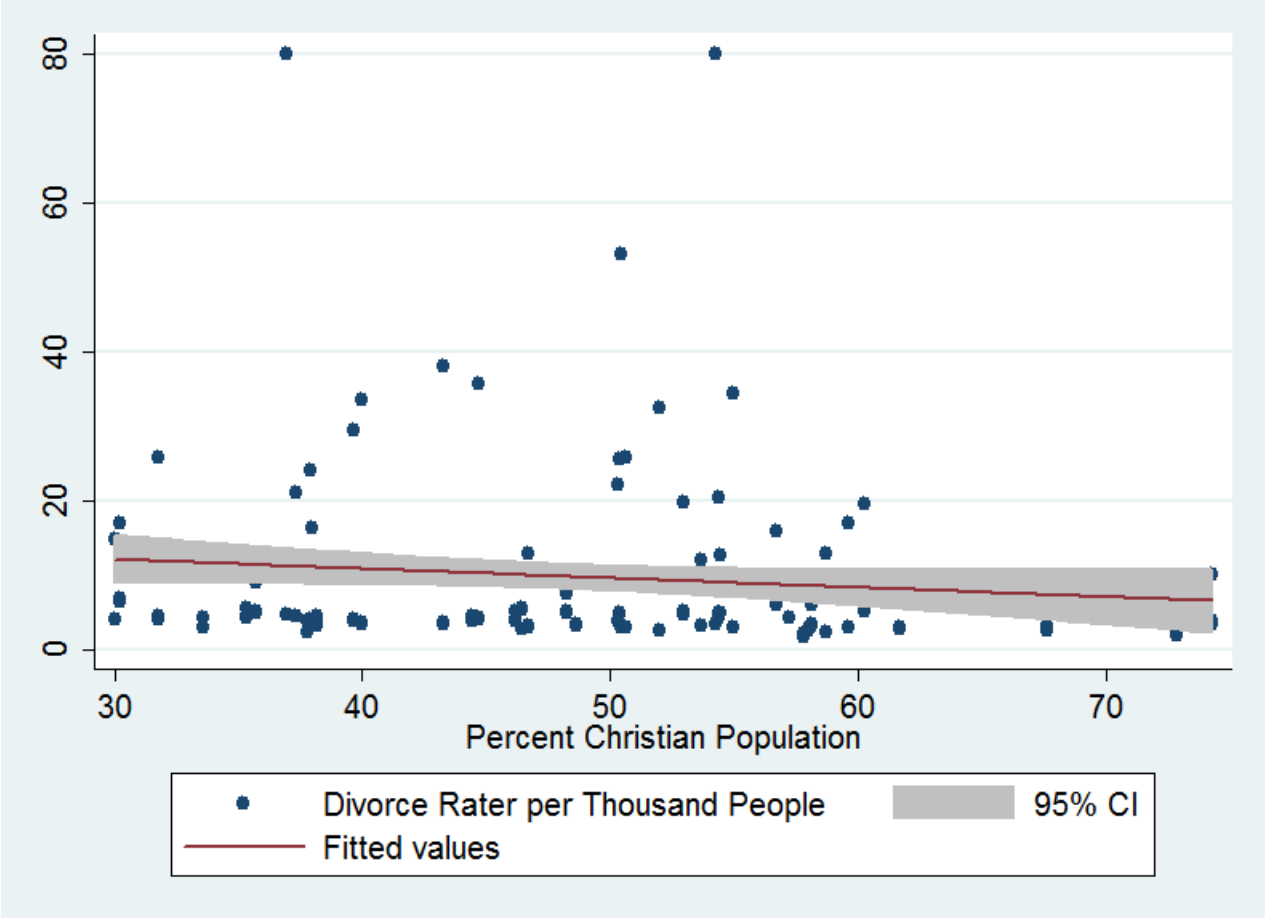
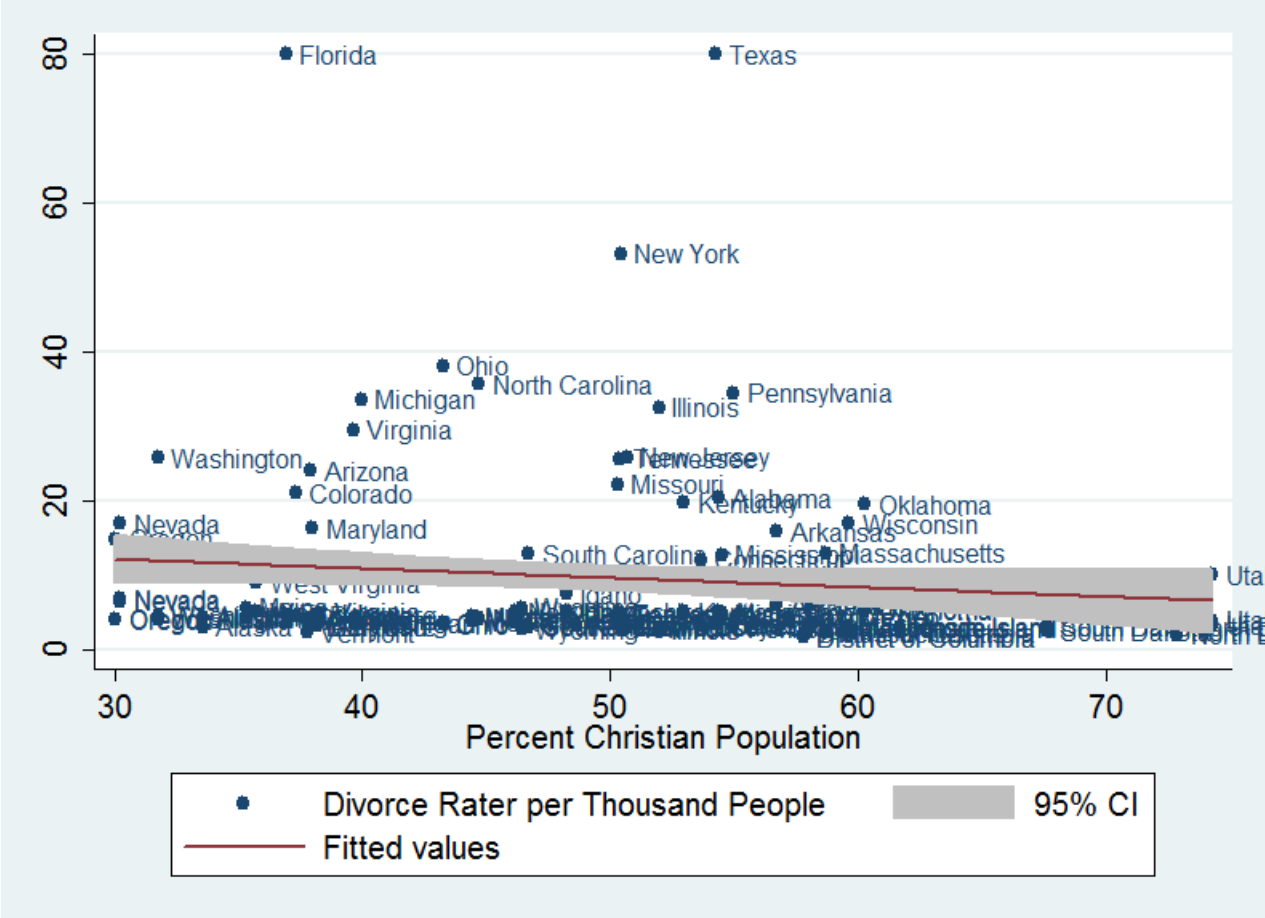


Figure 24: Estimated Effect with Confidence Intervals Canned Stata Package



References

- Arena, Phil. 2009. "Explaining International Politics."
URL: http://web.me.com/filarena/Site/Teaching_files/explainingIR.pdf
- Beck, Nathaniel and Jonathan N. Katz. 1995. "What to do (and not to do) with Time-Series Cross-Section Data." *The American Political Science Review* 89(3):634–647.
- Gujarati, Damodar N. 2003. *Basic Econometrics*. Fourth ed. New York, NY: McGraw Hill.
- Long, J. Scott. 1997. "Regression Models for Categorical and Limited Dependent Variables." *Advanced Quantitative Techniques in the Social Sciences* 7.
URL: <http://www.indiana.edu/~jslsoc/spost.htm>
- Teele, Dawn L. 2010. "The Stata Bible 2.0." Department of Political Science, Yale University.
URL: <http://statlab.stat.yale.edu/workshops/>
- Tomz, Michael, Jason Wittenberg and Gary King. 2003. "CLARIFY: Software for Interpreting and Presenting Statistical Results." *Journal of Statistical Software* 8.
URL: <http://gking.harvard.edu/files/abs/making-abs.shtml>
- UCLA. N.d. "Statistical Computing." UCLA Academic Technology Services.
URL: <http://www.ats.ucla.edu/stat/stata/default.htm>