

Advanced Regression Extracting Data Using Stata

Large Data Sets

In order to make datasets universally available, firms and government agencies which produce raw data often package them in machine readable files. These files, usually just ASCII text files or ASCII delimited text files, contain all the relevant information from a dataset for all observations. These data are often organized in large blocks of numbers, and if you could see these blocks, it would look like a large matrix of information. Typically, each row of the block is an individual observation and each column (or group of columns) represents different variables. This handout will walk you through the steps you need to complete to obtain raw data from a large data set, clean the data, and prepare it for analysis. This procedure will be especially useful for your Research paper.

Extracting Data

To extract data from an ASCII raw data file, you need to complete the following steps:

1. Pick the variables you intend to use in your analysis.
2. In the codebook (usually provided by the organization that created the dataset), find each column designation for each variable you have chosen.
3. Create a data dictionary file.
4. Run the data dictionary file in Stata.
5. Explore and check your data.
6. Create a file to manipulate and clean the data according to your needs.

Below, I discuss each step in turn.

Step 1: Pick your variables

Look at either the survey questionnaire or the dataset's codebook. A codebook is a physical listing of all variables in the dataset, and the codes for each variable. For example, if you were interested in a variable on race, the codebook would tell you where in the ASCII data file the race variable is located, what the codes recorded in the ASCII data file mean (1 = Asian, 2 = Hispanic, etc.), and the frequency of observations associated with each variable and answer category. The codebook is a great reference point to begin with, and a great place to check that you have correctly read in the data. Before you trove the codebook for variables, you should have a pretty clear picture of the model you wish to estimate and the variables that will appear in the model. Do not get too carried away with variable selection? What should you control for? What is your dependent variable? Keep it parsimonious.

Step 2: Variable locations in the raw ASCII data files

The codebook usually provides information on the location of each variable in the raw ASCII data file. Remember, data are organized as a large block of numbers. Each row of a data file is an observation, and each column represents a variable or a part of a variable. Thus, to be able to read data from an ASCII data file, you must tell Stata where to go to look for each variable. Stata also needs to be alerted to number of columns to grab in order to correctly create the variable list. Below is an example of a codebook entry from the NELS dataset.


```

_column(414) float byp45a %1f "Held back because of parental request"
_column(415) float byp45b %1f "Held back because of school request"
_column(416) float byp45c %1f "Held back because of other reason"
_column(417) float byp46a %1f "8th grdr repeated K"
_column(418) float byp46b %1f "8th grdr repeated 1"
_column(419) float byp46c %1f "8th grdr repeated 2"
_column(420) float byp46d %1f "8th grdr repeated 3"
_column(421) float byp46e %1f "8th grdr repeated 4"
_column(422) float byp46f %1f "8th grdr repeated 5"
_column(423) float byp46g %1f "8th grdr repeated 6"
_column(424) float byp46h %1f "8th grdr repeated 7"
_column(425) float byp46i %1f "8th grdr repeated 8"

}

```

Each data dictionary has two parts. First, the dictionary must tell Stata where the raw ASCII data file is located. This part is denoted by the command `dictionary` using `h:\nels92\stmeg.pri`. Second, it must tell Stata how to read that raw ASCII file to find the variables you are looking for. This is what the remainder of the dictionary file is communicating to Stata.

There are several variable types available in Stata. “Float” designates that the decimal is a floating decimal (we do not know beforehand where the decimal point will be located). Other designations include byte, integer, and string variables. A string variable is a letter or word rather than a number. For a string variable, you must let Stata know how long the string is. If you were to assign 15 characters to the string, you would enter `str15` rather than `float`. Use float and string only. After you load your data into Stata, type `compress` and Stata will automatically change variable types as appropriate. Assign the variable names given in the codebook, since you are likely to generate new variables and will want to save the best variable names for those.

For the column block designation, notice that each line starts with a column location, the type of variable (float in this case), the variable name (as stated in the codebook), and the length of the variable (as indicated by a percent sign, followed by a number, and an “f”). For the variable identified in the sample codebook above, the length of the variable is 2—covering two columns. Notice that the length designation in the data dictionary is “%2f”, telling Stata to take two columns of data for each observation, beginning with column 307. The final entry is the variable label.

There are some additional factors to note. The data dictionary column designation block begins with a “{” and ends with a “}”. The column designation starts with an underscore (“_”). Variable label names are contained in quotes.

Step 4: Running the Data Dictionary File in Stata

Once the data dictionary file is complete, you should execute it. To do that, open up Stata and type the following command:

```
infile using datadict.dct
```

where you supply the name and file location of the data dictionary in the place of *datadict.dct*. Be sure that the raw ASCII data file is accessible, that you point Stata to the correct location of the raw data file, and that your data dictionary has the correct file designation. If there is an error, Stata will let you know. If you have done everything correctly, Stata will list the set of variables in the “variable” box, as well as provide you with an indication of the number of observations in the dataset. This step can often cause people a lot of problems, and with it, stress. Stata is very particular about the proper construction of the dictionary. If there are any errors, Stata will not run the dictionary. Double-check it. Make sure the location of the raw data file is correctly specified in the dictionary. Make sure the location of dictionary is

correctly specified in the `infile` statement. Finally, make sure the various symbols (“%”, “_”, etc.) are consistently included in the dictionary.

Step 5: Explore Your Data

Congratulations! You have successfully read data into Stata. But this is just the beginning. Now is the time to go back to the codebook. With codebook in hand, summarize/tabulate every variable in the dataset. Check to see that the minimum and maximum values of the data match the values recorded in your codebook. If they do, tabulate each variable to see if the frequencies match those indicated in the codebook. Also, check that the number of observations in your data does indeed match that in the codebook. If anything is amiss, you should check your data dictionary column designations to make sure you are taking information from the correct columns.

Step 6: Cleaning Your Data and Creating Analysis Variables

Even though you have read in your data, it is still in a raw form. You now need to clean it up (e.g., create variables, throw out useless observations, etc.). To do this, create a data/variable generation file. This is just a `.do` file which will re-create all the necessary variables for you and all of the necessary steps you took along the way to create the final analysis sample. I HIGHLY recommend that you write a `.do` file for your data generation step. Creating variables interactively is fine, but you run the risk of forgetting exactly what you did. Plus, if someone wishes to verify your work, you could hand over the data generation files easily. Also, if you lose your data for some reason, you can simply re-run the data dictionary and data generation file to get everything back quickly. Another thing to remember is to meticulously document the variable generation file, taking note of each variable you create, why you are creating it, and what you did to create it. Such notes along the way will help to “jog” your memory if you need to re-visit the dataset months or years after the initial set of analyses. Remember: Getting your data in order is the hardest part of doing good research. As the saying goes: Garbage in, garbage out. Adequate time spent initially getting your data cleaned up and ready for analysis will pay enormous dividends when it’s time to analyze. Have fun!