STATA Hand Out[1]

STATA Background:

STATA is a Data Analysis and Statistical Software developed by the company STATA-CORP in 1985. It is widely used by researchers across different fields. STATA is popular because it provides everything you need for data analysis, data management and graphics.

STATA's latest version is version 12. Most commands in this hand-out work on all versions of STATA.

STATA on your computer:

To open STATA, click on the short-cut icon on your desktop which looks like this --  . This is the icon for STATA 8 (Exercise: whats the version of your STATA).

The STATA Window

When you open STATA, it will look like the picture below. As you can see, STATA always has four windows open:

1. The **Variables** window: this is the box at bottom left corner of the screen. **It shows a list of all of the variables in your dataset.**
2. The **Command** window: this is the window at the bottom right of the screen. **In this window, you type various commands to tell STATA what to do.**
3. The **Results** window: This is the large black window in the centre of the screen. **This window displays the results generated by the commands you have entered.**

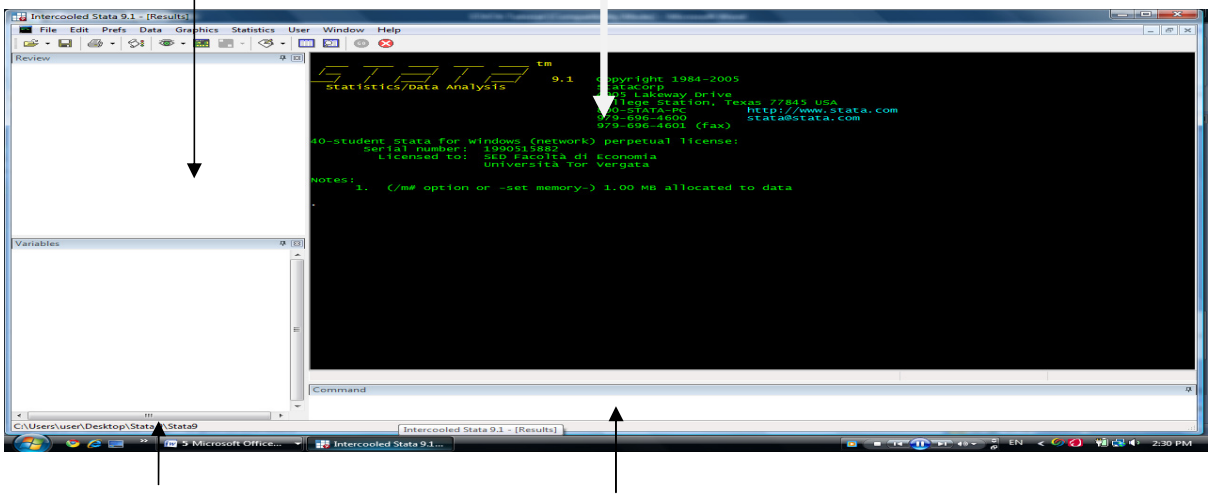
[1] The instructor wishes to thank ASER centre for access to its training notes. Some of the material in this handout has been sourced from there.

Abhiroop Mukhopadhyay, Assistant Professor, Indian Statistical Institute (Delhi): *abhiroop@isid.ac.in*

4. The **Review** window: This is the box in the top left corner. **<u>It shows a list of all the commands you have told STATA to do (i.e. everything you have typed in the command window).</u>** This window is useful if you want to repeat a command. Therefore, instead of retyping the command, you can just click on the command that you want to repeat. The command will then appear in the Command window and you can execute it from here.

STATA WINDOW

4. Review Window **3.** Results Window



1. Variables Window **2.** Command Window

Abhiroop Mukhopadhyay, Assistant Professor, Indian Statistical Institute (Delhi): *abhiroop@isid.ac.in*

A ZOOMED IN VIEW OF THE RESULT WINDOW (MONOCHROME)

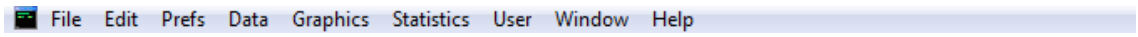
```
  ___  ____  ____  ____  ____ (R)
 /__    /   ____/   /   ____/
___/   /   /___/   /   /___/   11.1   Copyright 2009 StataCorp LP
  Statistics/Data Analysis             StataCorp
                                       4905 Lakeway Drive
     Special Edition                   College Station, Texas 77845 USA
                                       800-STATA-PC        http://www.stata.com
                                       979-696-4600        stata@stata.com
                                       979-696-4601 (fax)

Single-user Stata perpetual license:
```

YOU CAN SPOT THE VERSION NUMBER HERE TOO!!! (WHAT IS IT?)

The STATA Toolbar

Across the top of the screen you will see 2 toolbars. On each of these toolbars, there are a number of buttons that you can use while working with STATA. On the first toolbar, you will see buttons such as 'File' and 'Edit' which should already be familiar to you from programs like MS Excel. If you click on any of the buttons it will open a menu of options. Other buttons on this toolbar are not the same as in MS Excel, for example the 'Data' 'Graphics' and 'Statistics' buttons (picture is below). As you work with STATA, you will become familiar with these buttons

STATA Toolbar (top): THESE MAY VARY VERSION TO VERSION BUT NOT MUCH



For example, STATA 11 does not have "Prefs" but this functionality is subsumed into "Edit"

Abhiroop Mukhopadhyay, Assistant Professor, Indian Statistical Institute (Delhi): *abhiroop@isid.ac.in*

For now on your STATA screen, focus on the second toolbar that is below these buttons. This toolbar contains a series of short-cut buttons (picture is below). From left to right, these buttons are as follows:

STATA Toolbar (Short-Cut Buttons: THESE TOO VARY VERSION TO VERSION

1. **Open File:** clicking on this will allow you to **open a data set**
2. **Save:** clicking on this will allow you to **save the data set you are working with**
3. **Print:** clicking on this will **print all of your output from the results window**
4. **Log:** This button creates a 'log', or **record of all your commands and results** (more on this later)
5. **Viewer:** Clicking on this **button opens a fifth window, called the Viewer window. The viewer window is used for many things: looking at STATA help files, viewing results you have saved in a log file, or browsing through STATA documents online**
6. **Results:** This button **brings the results window to the front**
7. **Graph:** If you have made a graph in Stata, clicking on this button **will bring the graph window to the front.**
8. **Do File Editor:** This button **opens the Do File editor. In this window, you can write a series of commands for Stata to do and then run them all together. STATA will do all of the commands continuously in the order you have written them** (more on this later).
9. **Data Editor:** This button **opens the data editor. You will see the data displayed just like an MS Excel spreadsheet. Within this window you can make changes to data.**
10. **Data Browser:** This button **opens the data editor. Once again, you will see the data displayed just like an MS Excel spreadsheet. Within this window you *cannot* make changes to data.**
11. **Go:** Sometimes the results of a command take up more space than the STATA results window can show. When this happens, STATA pauses

as soon as the results window fills up. **To tell STATA to show the rest of the results, click on the 'Go' button.**

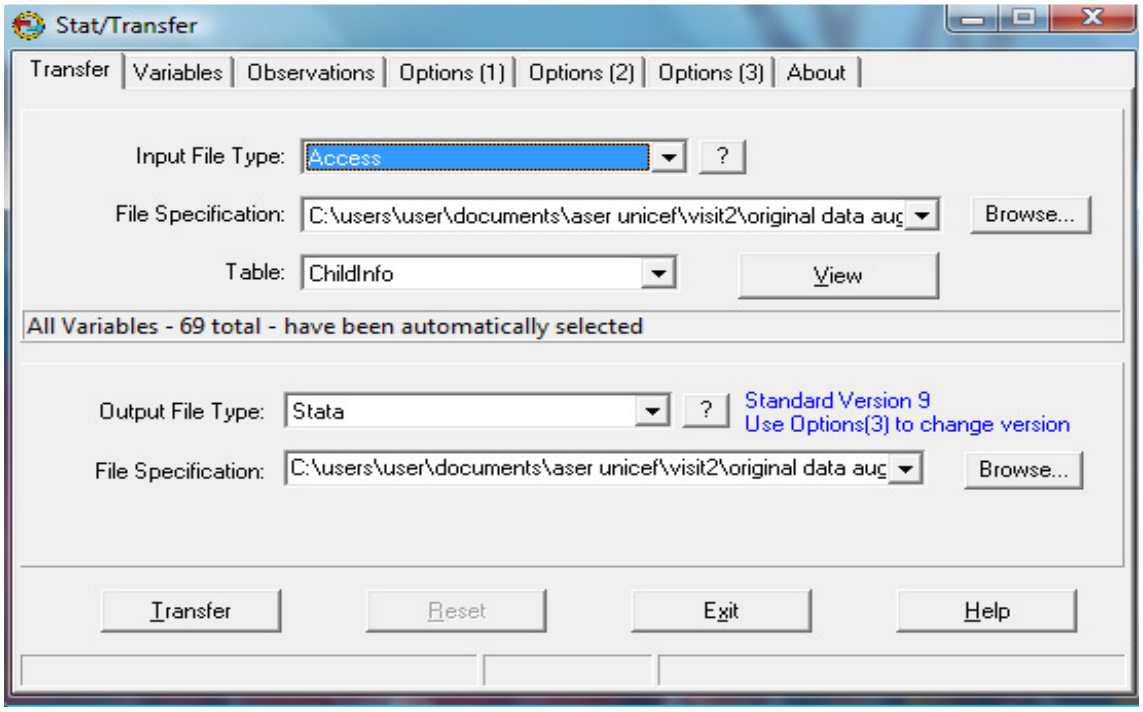
12. **Break:** Sometimes the results of a command take up more space than the results window can show, but you will not want to see the rest of the results. **Therefore, to tell STATA to stop executing the command, you can click on the "Break" button.**

Entering Data into STATA from Other Programs

STATA saves datasets in a file type called **.dta.** STATA cannot open data in any other format directly. So, for example, if your data is in Excel (.xls) or some other program, you will not be able to open it directly using STATA.

To open data from other formats use a simple format convertor called STAT Transfer.

A SAMPLE SHOT OF STAT TRANSFER



Opening STATA Data Files

Once your data is in STATA format (.dta) you can open it in STATA and begin using it. When opening data, you should always go through the following steps.

1. Set STATA's Memory

Opening a dataset requires STATA to use some amount of your computer's memory. Bigger datasets need more memory to open. Running commands requires more memory again. **When you first open STATA, it will be set to use only 1 megabyte of your computer's memory.** However, this amount may be too small to open up big datasets and run commands on them. **Therefore, it is very important that before you open any data set in STATA, you tell STATA how much of your computer's memory it can use to open and process data.**

Small data sets containing only a few observations require less memory than larger data sets.

type the following into the STATA command window and press enter.

set mem 1m

You could set the memory used to any size you like by changing the number. For example, later when you work with a large data set which needs 100 megabytes of memory to open. Before opening this datasets, you should type:

set mem 100m

Note: It is not possible to re-set the memory once a data set has been opened. You must clear the data in STATA's memory first. Further, if you are not sure how much memory to use, check the size of your data file, and set the memory to a number bigger than this.

Abhiroop Mukhopadhyay, Assistant Professor, Indian Statistical Institute (Delhi): *abhiroop@isid.ac.in*

RESULTS WINDOW

```
. set mem 100m

Current memory allocation

    settable       current value      description        memory usage (1M = 1024k)
-------------------------------------------------------------------------------
set maxvar          5000          max. variables allowed          1.947M
set memory           100M         max. data space               100.000M
set matsize           400         max. RHS vars in models         1.254M
                                                                ----------
                                                                  103.201M
```

2. Start a Log File

The next step is to start a 'log' file. **A log file records any commands that you type into STATA, as well as the output that these commands generate. The log file is a record of both your input and your output.** This is important as you can only view the last few commands and results in the results window on your STATA screen, and not all of the commands you have executed. Once you open a log file, all of your commands and results are automatically saved in this file. You can open the file again at some later stage and review everything you have done. You should always open a log file before beginning a STATA session.

EASIEST METHOD: Click File > Log > Begin ... at this stage many versions will ask you for a file name... the file is stored as **.smcl** (let us store the file as *nss_session 1* in the subdirectory *NSS\log*

METHOD 2: Click the appropriate short cut icon from STATA tool bar.

```
. log using "C:\Users\admin\Desktop\NSS\log\nss_session 1.smcl"
-------------------------------------------------------------------------------
      name:  <unnamed>
       log:  C:\Users\admin\Desktop\NSS\log\nss_session 1.smcl
  log type:  smcl
 opened on:  18 Oct 2011, 10:15:45
```

METHOD 3: TYPE *log using "C:\Users\admin\Desktop\NSS\log\nss_session.smcl"* IN THE COMMAND WINDOW (You will ofcourse have to change the path name in this command)

As you keep working on STATA, the log file captures every action that you are taking. If you would like to close your log file **permanently** in the middle of your STATA session - you can do this by typing the following command into the STATA command window and pressing enter;

log close

If you would like to close your log file ***temporarily*** in the middle of your STATA session - you can do this by typing the following command into the STATA command window and pressing enter;

log off

If you then want to turn on your log file once again - you can type the following command into the STATA command window and pressing enter;

log on

When you finish your STATA session, the log records ends as well.

Viewing Log File:

File > *Log* > *View* > *Browse*

Exercise

1. Log begin nss_session 1

2. Type *set mem 100m* in Command Window

3. Type *log off* in Command Window

Abhiroop Mukhopadhyay, Assistant Professor, Indian Statistical Institute (Delhi): *abhiroop@isid.ac.in*

(From here on, when I say Type, I mean Type in Command Window unless otherwise specified)

4. Type *set mem 1m*

5. Type *log on*

6. Type *set mem 50m*

7. Type *log close*

8. View log file.... What don't you see?

Loading Data into STATA

Once you have set the memory and opened a log file, you can open your dataset. Data is opened the same way as opening a file in MS Word or Excel:

METHOD 1: just click on File on the top right of the STATA screen. From the drop down menu, click on Open and locate the file you want.

METHOD 2: You can also click the 'open' icon referred to earlier on the far left of the STATA toolbar and locate the file you want.

METHOD 3: *use "C:/XX/XXX.dta", clear*

clear*: clears all previous data in the computers memory.*

.....End of Lecture 1.....

Exploring Data:

THREE METHODS:

1. TOGGLE THE TOOLBAR KEYS

2. GIVE COMMANDS IN THE COMMAND PROMPT

3. WRITE A PROGRAM FILE WITH ALL THE COMMANDS.

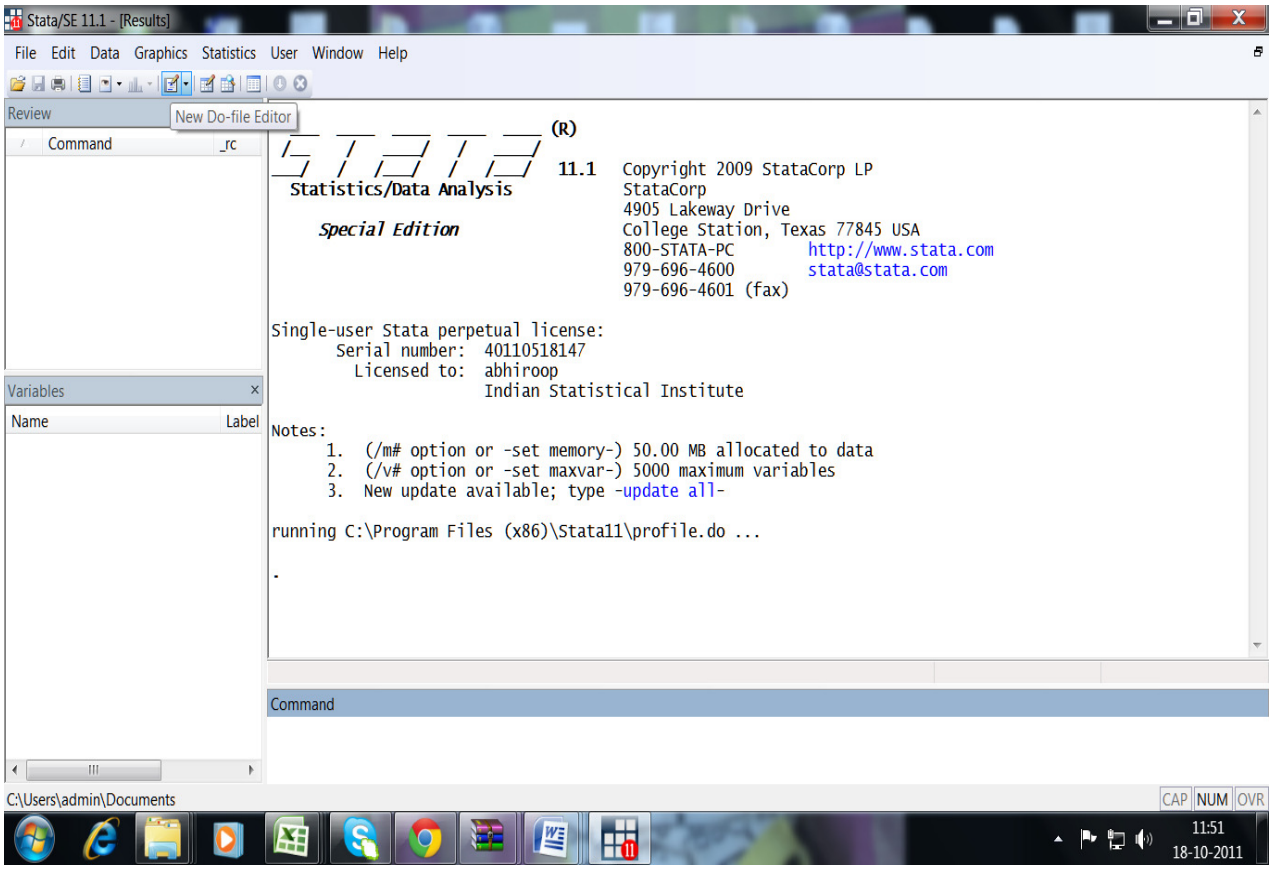
For this talk: we use method 3.

Do files:

- **A do-file is a program that you write for STATA.** It is called a 'do-file' because it is a file that is saved with the suffix '.do'.
- **Essentially, a do-file is a file in which you can write a list of commands that you want to run in STATA. When you are finished writing your commands, you can run the do-file and STATA executes each command in the do file in the order that you have written them**.
- **Do-files are a strongly recommended way of working in STATA.** This is because they are permanent records of all the commands and steps that you used to change/manipulate your original dataset and the results thereof.
- Saved do-files can be run at a later time to replicate the steps you took while working with your dataset.
- Do-files can also be shared with other people working on the same dataset quite easily.
- You can even insert comments into your do-file to help other people follow your thought process.

How to create a do file:

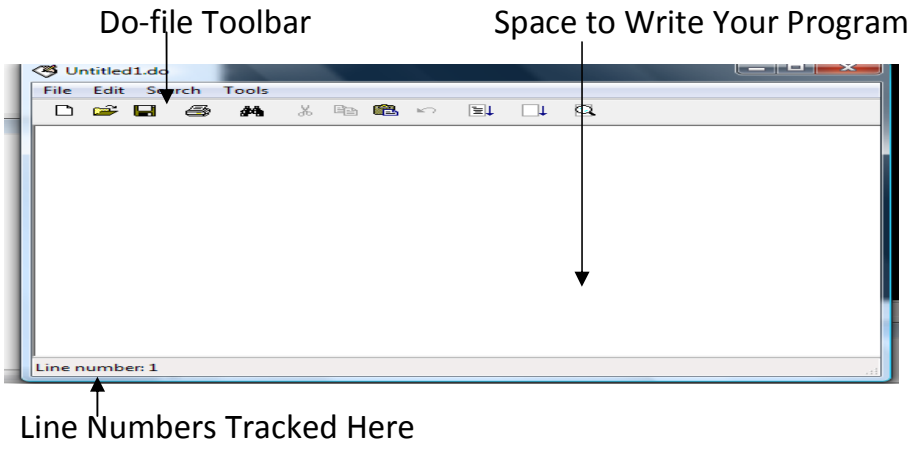
Take your cursor and check the toolbar... one of them will say "New do file Editor"...



The Do-file Window and Toolbar

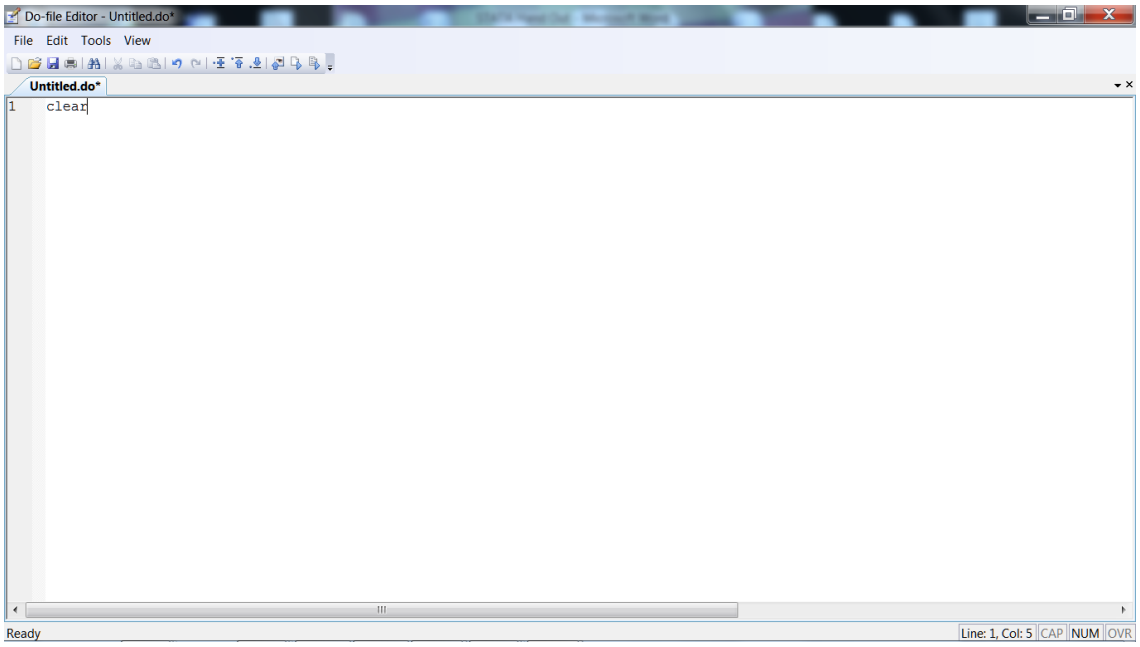
You will see a window open as shown in Picture below (Again different versions may be different but same idea)

Screen shot of a Do-File: STATA 9



Abhiroop Mukhopadhyay, Assistant Professor, Indian Statistical Institute (Delhi): *abhiroop@isid.ac.in*

Screen shot of a Do-File: STATA 11



Line Numbers Tracked Here

The do-file window in the picture closely resembles an empty Notepad document in MS Windows. You can type your program i.e. commands/comments on the white space provided in the window.

At the bottom of the window, you will see the text 'line number: 1'. As you write your commands and steps in the do-file, you will see the count for number of lines increase.

On the top of the do-file window you can see buttons like 'File' 'Edit' 'Search' and 'Tools'. You should already be familiar with these buttons from programs like Notepad and MS Word.

Abhiroop Mukhopadhyay, Assistant Professor, Indian Statistical Institute (Delhi): *abhiroop@isid.ac.in*

On the tool bar: find these two icons in the end.

1.  **Do Current File: <u>If you select a few lines within your do-file and click on this button, STATA executes the commands for those selected lines only</u>. (in later versions of STATA: "Execute Selection")**
2.  **Run Current File:** Clicking on this button **<u>runs the entire do-file you have writing without interruption</u> (in later versions of STATA: "Execute Selection Quietly (run)")**

For this presentation, we will click "Do Current File Only" (in later versions of STATA: "Execute Selection")

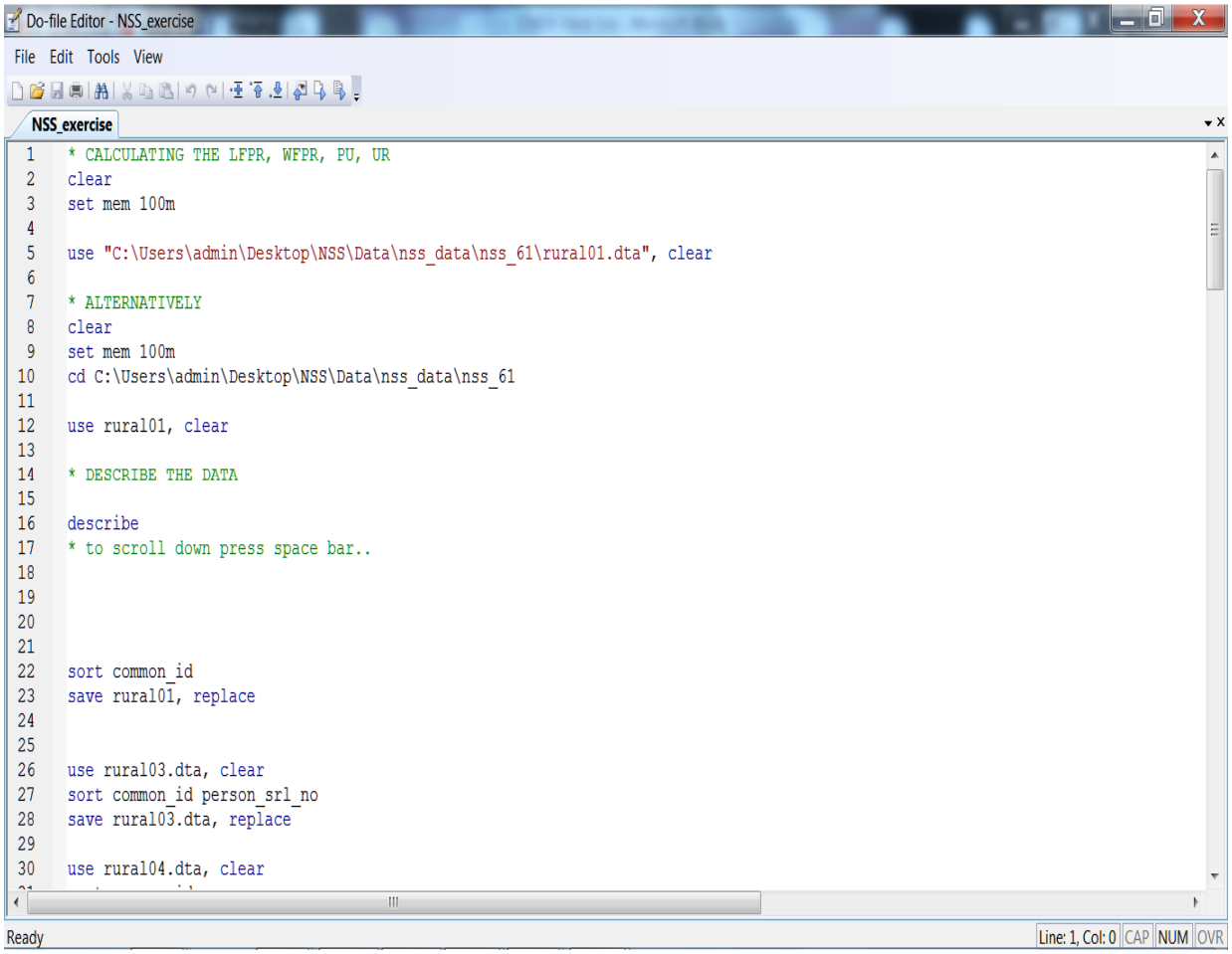
To see how it works, let us open an existing do file

Open existing do file.

METHOD 1: File > OPEN > (this will open a file browse window)... choose Do File format
Click File Name

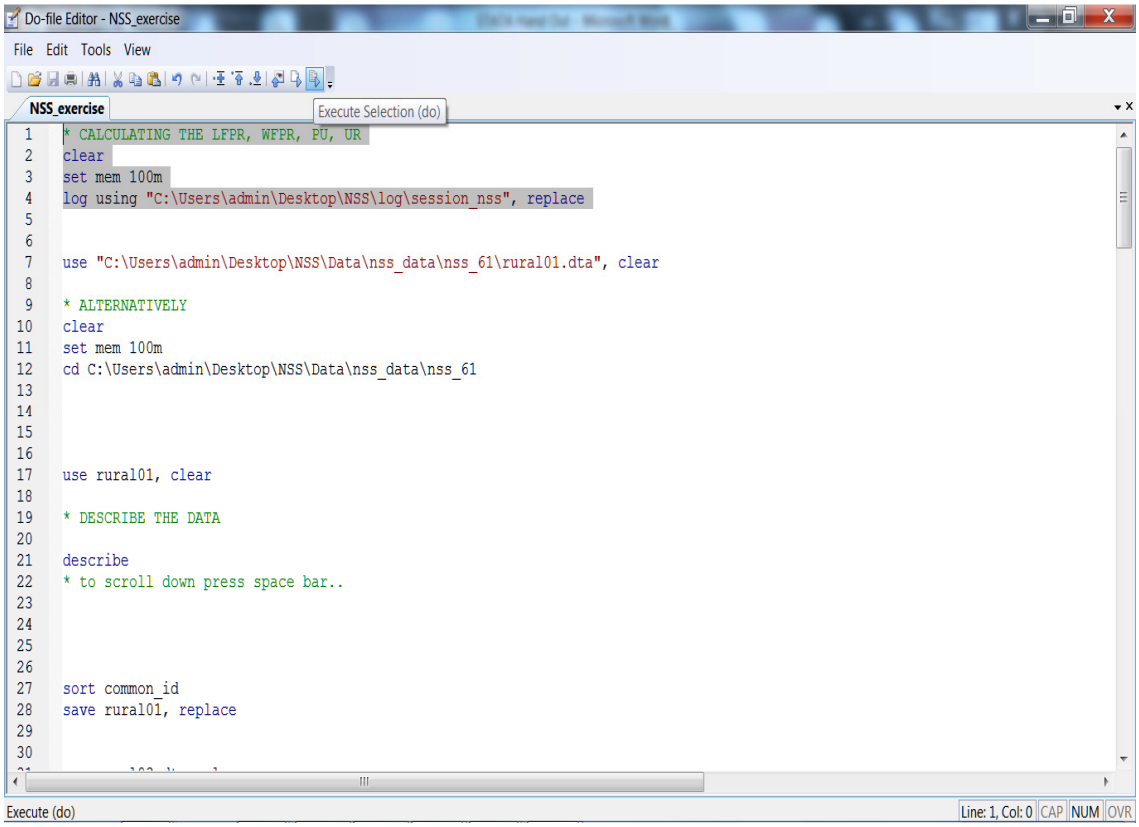
METHOD 2: Click the "Do file" Icon on short cut toolbar and File open...

Let us click and go to the File NSS_exercise.do in folder ../Do_file



To Run Selected lines, (lets say the first three lines)... block the three lines and click "Do Current File"(in later versions of STATA: "Execute Selection") from the toolbar.

Abhiroop Mukhopadhyay, Assistant Professor, Indian Statistical Institute (Delhi): *abhiroop@isid.ac.in*

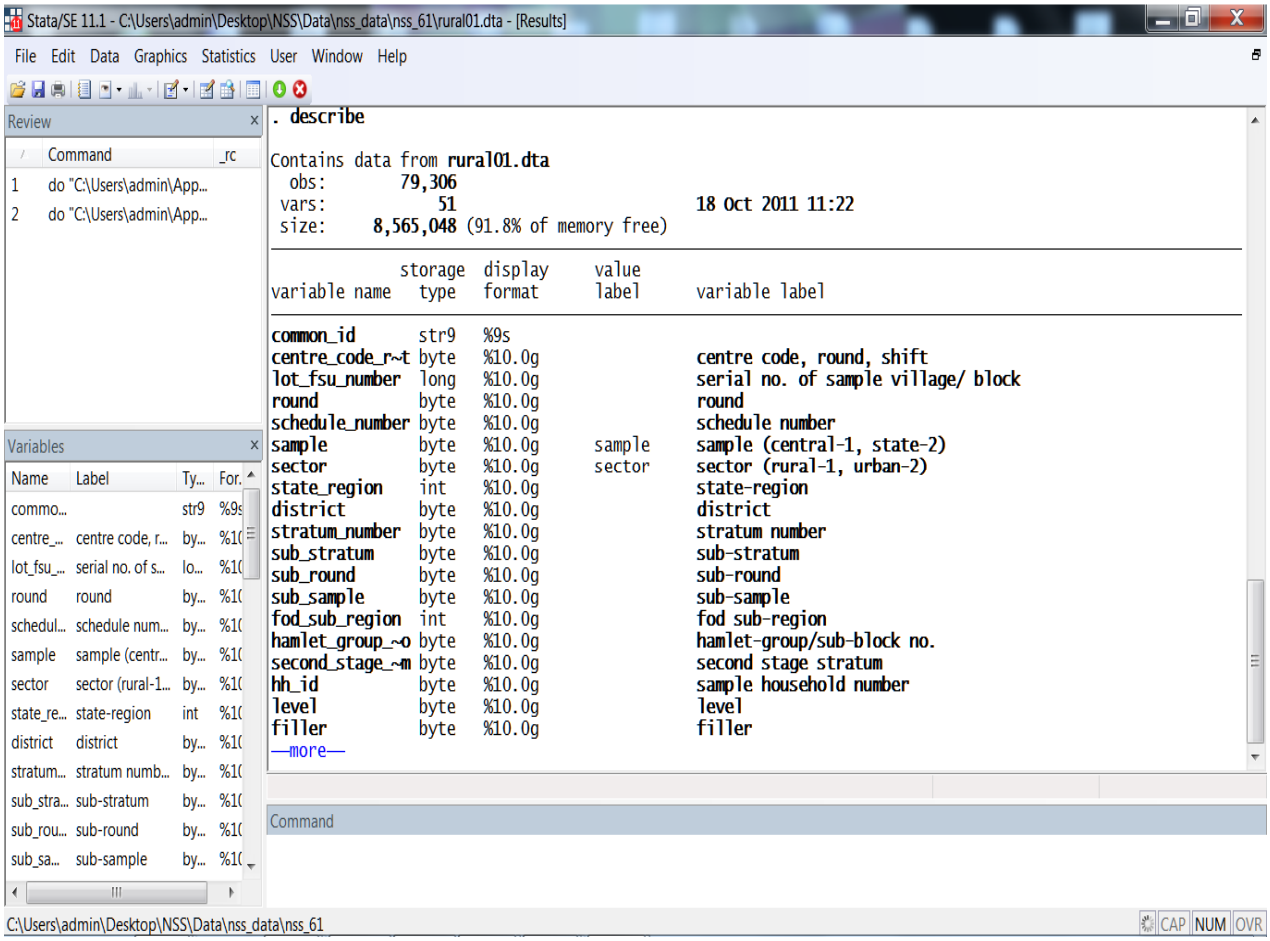


* Any line beginning with an asterix(*) is a comment in a do file.

1. Command: *describe*

Purpose: <u>The *describe* command helps to describe the variables in your dataset</u>. It provides details on the variable name, whether the variable is numeric or string, and details of any labels[2] attached to the variables and its values.

[2] 'labels' refer to descriptions attached to the variable and values of a variable. This will be dealt with in more detail in Chapter 7.



- The first column in the results window gives the names of the variables.
- The second column shows the “storage type”. This means the way in which Stata stores the variable. STATA stores numeric variables as byte, int, long, float or double (depending on how big the numbers in a variable are). STATA stores string variables as str.
- The third column shows how much space given to it and what type of variable it is declared. For example common id : %9s says it is a variable that has been given nine spaces and its a string
- Fourth variable we will come to later
- Fifth variable are labels given to data set (In the case of NSS data, these are given during the time of extraction. we come to this later)

Notice -More- at the end of the page. You have to press space bar for it to scroll down further.

You might want to describe only one variable in the dataset, rather than all of them. You can do this by typing *describe*, followed by the name of the variable

For example : describe sub_round

```
. describe sub_round

              storage  display     value
variable name   type   format      label      variable label
------------------------------------------------------------------------------
sub_round       byte   %10.0g                 sub-round
```

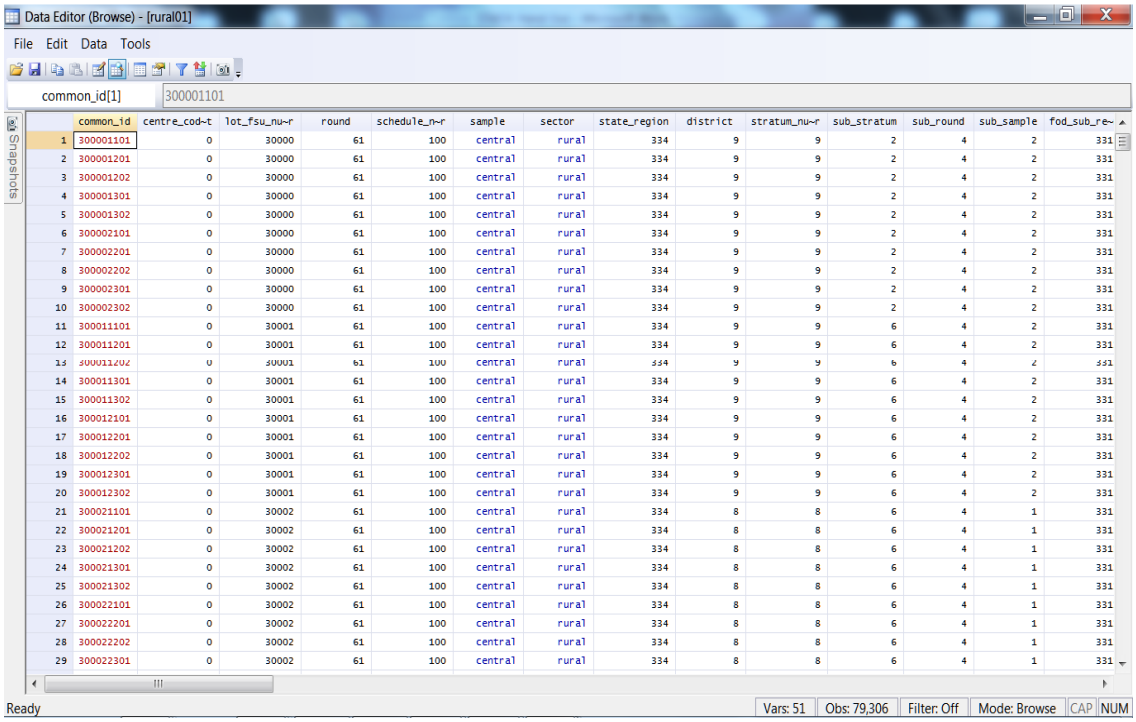
2. Command: *lookfor*

Purpose: <u>This command helps you to find a variable in your dataset.</u> You can use this command when you are not sure of the variable's name.

```
. lookfor state

              storage  display     value
variable name   type   format      label      variable label
------------------------------------------------------------------------------
sample          byte   %10.0g      sample     sample (central-1, state-2)
state_region    int    %10.0g                 state-region
state           byte   %17.0g      state_61
```

3. Command: *browse*

Purpose: <u>This command is used when you want to display data in STATA's data browser window.</u> This command is useful if you want to take a closer look at the raw data. When you use the browse command you will see the data displayed just like an Excel sheet.



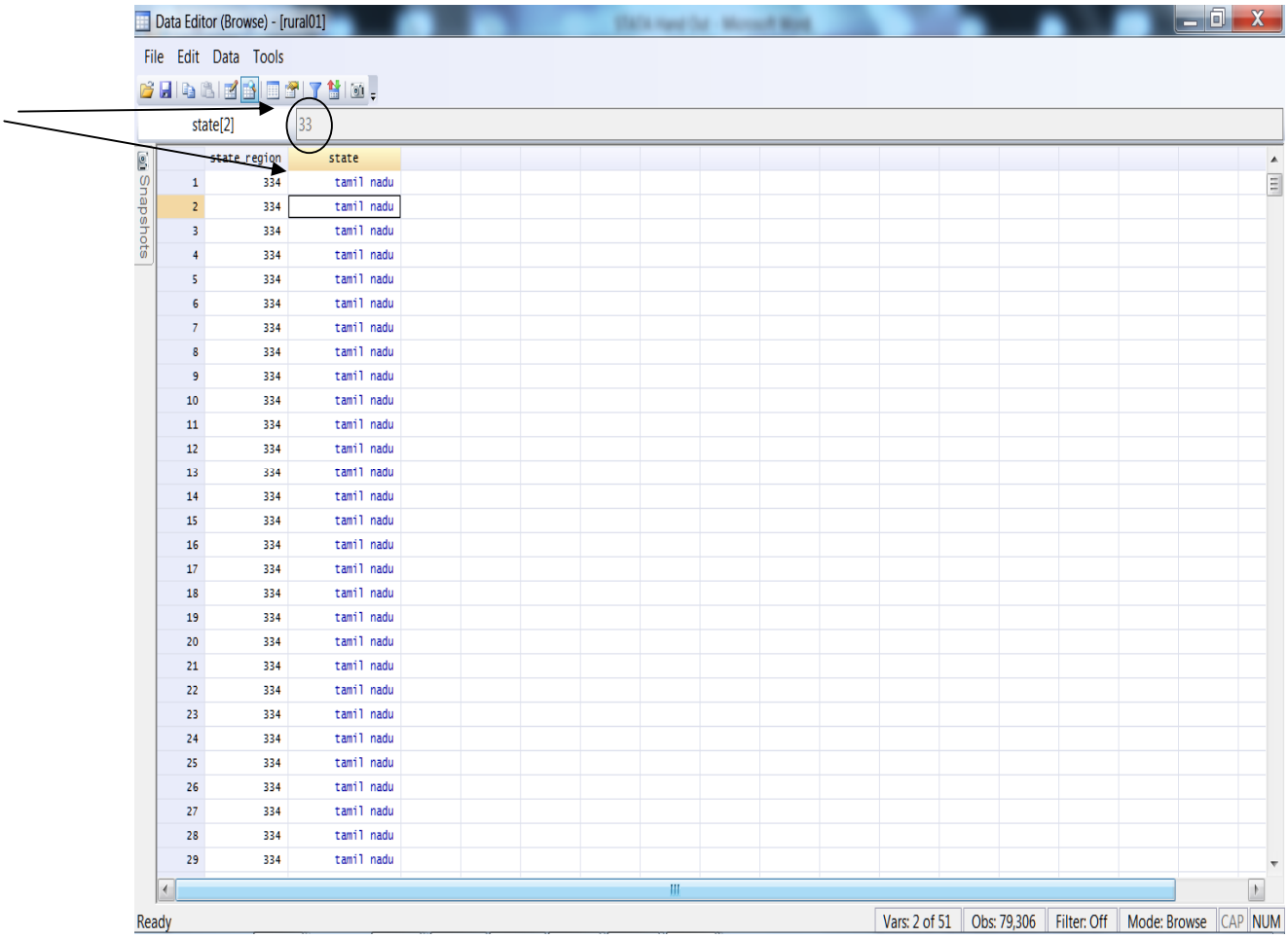
	common_id	centre_cod~t	lot_fsu_nu~r	round	schedule_n~r	sample	sector	state_region	district	stratum_nu~r	sub_stratum	sub_round	sub_sample	fod_sub_re~
1	300001101	0	30000	61	100	central	rural	334	9	9	2	4	2	331
2	300001201	0	30000	61	100	central	rural	334	9	9	2	4	2	331
3	300001202	0	30000	61	100	central	rural	334	9	9	2	4	2	331
4	300001301	0	30000	61	100	central	rural	334	9	9	2	4	2	331
5	300001302	0	30000	61	100	central	rural	334	9	9	2	4	2	331
6	300002101	0	30000	61	100	central	rural	334	9	9	2	4	2	331
7	300002201	0	30000	61	100	central	rural	334	9	9	2	4	2	331
8	300002202	0	30000	61	100	central	rural	334	9	9	2	4	2	331
9	300002301	0	30000	61	100	central	rural	334	9	9	2	4	2	331
10	300002302	0	30000	61	100	central	rural	334	9	9	2	4	2	331
11	300011101	0	30001	61	100	central	rural	334	9	9	6	4	2	331
12	300011201	0	30001	61	100	central	rural	334	9	9	6	4	2	331
13	300011202	0	30001	61	100	central	rural	334	9	9	6	4	2	331
14	300011301	0	30001	61	100	central	rural	334	9	9	6	4	2	331
15	300011302	0	30001	61	100	central	rural	334	9	9	6	4	2	331
16	300012101	0	30001	61	100	central	rural	334	9	9	6	4	2	331
17	300012201	0	30001	61	100	central	rural	334	9	9	6	4	2	331
18	300012202	0	30001	61	100	central	rural	334	9	9	6	4	2	331
19	300012301	0	30001	61	100	central	rural	334	9	9	6	4	2	331
20	300012302	0	30001	61	100	central	rural	334	9	9	6	4	2	331
21	300021101	0	30002	61	100	central	rural	334	8	8	6	4	1	331
22	300021201	0	30002	61	100	central	rural	334	8	8	6	4	1	331
23	300021202	0	30002	61	100	central	rural	334	8	8	6	4	1	331
24	300021301	0	30002	61	100	central	rural	334	8	8	6	4	1	331
25	300021302	0	30002	61	100	central	rural	334	8	8	6	4	1	331
26	300022101	0	30002	61	100	central	rural	334	8	8	6	4	1	331
27	300022201	0	30002	61	100	central	rural	334	8	8	6	4	1	331
28	300022202	0	30002	61	100	central	rural	334	8	8	6	4	1	331
29	300022301	0	30002	61	100	central	rural	334	8	8	6	4	1	331

You cannot make changes to the data from this window. In order to close the browser window - click on the Cancel button on the top right of the screen. 

You can browse selective variables: For example, if you want to see only state_region and state: then

browse state state_region

Abhiroop Mukhopadhyay, Assistant Professor, Indian Statistical Institute (Delhi): *abhiroop@isid.ac.in*



NOTE: LOOK AT THE ARROW AT 33. While the browser shows Tamil Nadu, the bar on top which shows what cell you are looking at shows 33. This because while extracting (or later) we can put labels to values. So the data is numeric but it shows as label when you see the data.

Digression: 'codebook' Command

Purpose: <u>The command codebook can be used to examine full details of the variables (including the variable labels, code values and labels).</u>

For example Religion values have been labelled in the data: Examine the labels using the statement

codebook religion

```
. codebook   religion

-------------------------------------------------------------------------------
religion                                                              religion
-------------------------------------------------------------------------------

                  type:  numeric (byte)
                 label:  religion

                 range:  [1,9]                        units:  1
         unique values:  8                    missing .:  9/79306

            tabulation:  Freq.   Numeric  Label
                        60741         1  Hinduism
                         8486         2  Islam
                         5655         3  Christianity
                         2221         4  Sikhism
                           80         5  Jainism
                         1002         6  Buddhism
                            1         7  Zoroastrianism
                         1111         9  others
                            9         .
```

4. Command: *inspect*

Purpose: <u>The *inspect* command provides a quick summary of a numeric variable</u>. It reports the number of positive, negative and zero values, and the number of unique and missing values contained within the numeric variable. This command is useful to get familiar with unknown data quickly and easily.

```
. inspect state

state:
------

|      #                          Negative        -        -        -
|      #   #   #                  Zero            -        -        -
|      #   #   #                  Positive    79306    79306        -
|      #   #   #   #              ----------  -------  -------  -------
|  #   #   #   #   #              Total       79306    79306        -
|  #   #   #   #   #              Missing         -
|-------------------------------  ----------  -------
1                          35                  79306
  (35 unique values)                Number of Observations
                                    Total    Integers    Nonintegers

    state is labeled and all values are documented in the label.
```

5. Command: *sort*

Purpose: <u>The *sort* command arranges observations in the dataset into ascending order. The dataset is sorted based on the values of the variables specified in the command. The sort command is especially useful when merging two separate data-files.</u>

Abhiroop Mukhopadhyay, Assistant Professor, Indian Statistical Institute (Delhi): *abhiroop@isid.ac.in*

One variable sorting: *sort lot_fsu_number*

Sorting by Two variables: *sort lot_fsu_number hh_id*

6. Command: *save*

Your data-file can be saved in STATA in two ways. These are as follows;

Method 1: You can use the save button on the STATA toolbar  . If you click on this button, Stata will ask you if you would like to 'overwrite the existing file'. i.e. Stata is checking if you want to save your dataset in its current form. By clicking on yes you will then replace the existing data file with the new one.

Method 2: The second way to save your data-file is by typing the command *'save'* followed by a file-path in the STATA command window. The syntax of this command is as follows;

save "c:\<foldername>\<sub-foldername>\<filename.dta>", replace

By saying 'replace' at the end of the command, you are telling Stata to over-write an existing data file in your computer's memory by the same name with the new file. **Please note the position of the symbols " : \ , Incorrect placement of these symbols will not allow your command to work**.

If you collaborator (other sub centres) have older version of STATA, it is recommended, you use the statement

saveold "c:\<foldername>\<sub-foldername>\<filename.dta>", replace

saveold saves the data as a version 9 STATA file.

Conditional Statement

If When using the qualifier 'if' you will need to specify appropriate logical operations. Logical operators are represented by special symbols in STATA. The symbol for each of the operators in STATA and its meaning has been explained below.

= = A double equal to sign is used to symbolize the operation **'equal to'**

~ = A 'tilde' symbol followed by a single equal to sign is used to symbolize the operation **'not equal to'**

> A greater than sign is used to symbolize the operation **'greater than'**

< A less than sign is used to symbolize the operation **'less than'**

> = A greater than sign followed by a single equal to sing is used to symbolize the operation **'greater than and equal to'**

< = A less than sign followed by a single equal to sing is used to symbolize the operation **'less than and equal to'**

& You should already be familiar with this sign. It symbolizes the operation **'and'**

| This sign symbolizes the operation **'or'**

Example : *browse if state==33 & sub_round>=2*

This will show you the data for state code 33 and when sub round numbers are 2, 3 and 4.

----------End of Lecture 2----------

Renaming Variables:

Purpose: Often we receive the data from the data-entry operators we find that sometimes we have to rename variables in order to make them easy to use and identify. The command that is used to do this in STATA is called **rename**. The syntax for this command is as follows;

rename <variable name> <new variable name>

For example: you want to rename fsu numbers as village.

rename lot_fsu_number village

Dropping/Keeping Variables and Observations

When working with data, you will sometimes find that there are variables or observations in your dataset that you do not need. In such a situation the commands **drop or keep** can be used in STATA.

6.4.1 The Drop/Keep Commands

Purpose: The command 'drop' *deletes* variables or observations from a dataset.

The command 'keep' works the in the same way as drop, except in this case you have to specify the variables or observations that you want to *keep*, rather than the variables or observations you want to delete.

6.4.2 Dropping/Keeping Variables

If you want to drop a single variable or a list of variables, the syntax to do this is as follows;

drop <variable list>

keep <variable list>

For example:

drop social_group

will drop social group from the data set

keep common_id mpce_30_days

will give you a data set with just those two variables.

NOTE:

If you save the file at this stage, then the file will only have common_id and mpce_30_days.

On the other hand, if you do not save the data set, then the last saved version will still exist and the changes you have made to the data set will be lost as

soon you close the session. (to access it you have to open that data set "rural01" in the class example again.

Once you drop or keep, the variables dropped will not be available in the session (unless the original data set rural01.dta is again loaded).

Dropping /Keeping Observations

If you want to delete observations from a dataset, you can use the commands drop/keep with qualifier *if* and the relevant operators.

For example if we want observations for only Hindus, then

drop if religion~=1

OR

keep if religion==1

Multiple Qualifiers can be used: So data for only Hindu Schedule Castes:

keep if religion==1 & social_group==2

Generating a New Variable

When working with data in STATA very often you will find that you need to generate a new variable to help your analysis. As far as possible, when using STATA you should always strive to work with a dataset that has primarily numeric variables. Therefore, if required you should convert string variables into numeric variables[3]. **Numeric can either be discrete, continuous, categorical or dummy.**

Command: *gen*

Purpose: In STATA the command to generate a new variable is known as generate or *gen* for short. The syntax for this command is as follows;

[3] This tutorial will not deal with converting string variables into numeric. For further information on this, you can visit the UCLA STATA website link - http://www.ats.ucla.edu/stat/stata/faq/destring.htm

gen <variable name> = <value of the variable>

Note: When generating the variable you only need to use the equal to sign once.

From the above you can see that when generating new variables you must always specify the name of the variable and its value.

In the dataset given, let us generate a variable called round and give it the value 61 as this is data from the 61st round.

```
. use rural01, clear

. * Generating a variable
.
. gen round=61
round already defined
r(110);

end of do-file

r(110);
```

There is an error message because the variable round already exists (how will you check that it already exists?)

Lets say that the variable round has a wrong value. Data inputter have input the value 55 instead of 61. what would you do?

drop round

gen round=61

NOTE: all newly generated variables come at the end of the data set... SCROLL DOWN on the variables list window to find the new variable round (check its indeed 61 by visual inspection)

Abhiroop Mukhopadhyay, Assistant Professor, Indian Statistical Institute (Delhi): *abhiroop@isid.ac.in*

Another context: Let us assume that NSS has reported the total household consumption incorrectly as MPCE in its data set. You realize this and want to create a variable mpce_corr that divides this incorrect MPCE by household size to get the correct mpce.

gen mpce_corr= mpce_30_days/ hh_size

Command: ***replace***

Another way to replace the value of 55 by 61 in the original data set (one of the examples above) is just replace the value of 55 for the variable *round* to 61.

replace round=61

Combining gen and replace with IF

Example... I want to create a variable which indicates if a household has mpce less than Rs 800 (poor?).

gen poor=1 if mpce_corr<=800

But what of those that have mpce_correct above 800. STATA puts them as . s by default. We may want to replace them by 0

replace poor=0 if mpce_corr>800

Analyzing Data:

** For the moment abstract from the idea that there are multipliers.*

The 'sum' Command

1. The Simple Sum Command: The sum command calculates and then displays a variety of summary statistics for a dataset. **<u>If you type the command sum by itself (i.e. without specifying any variables) STATA calculates summary statistics for *all* the variables in the dataset.</u>**

Abhiroop Mukhopadhyay, Assistant Professor, Indian Statistical Institute (Delhi): *abhiroop@isid.ac.in*

Example: To illustrate type the following command into the STATA command window and press enter;

sum

Output: You will see the following output appear on the STATA screen; (Please note that this is a snapshot of the entire output).

'sum' Output

```
. summ
```

Variable	Obs	Mean	Std. Dev.	Min	Max
common_id	0				
centre_cod~t	79306	0	0	0	0
lot_fsu_nu~r	79306	35107.33	2865.672	30000	39999
round	79306	61	0	61	61
schedule_n~r	79306	100	0	100	100
sample	79306	1	0	1	1
sector	79306	1	0	1	1
state_region	79306	181.2413	93.70666	11	351
district	79306	14.33463	12.9517	1	70
stratum_nu~r	79306	14.4017	13.14666	1	99
sub_stratum	79306	5.22567	4.341367	1	38
sub_round	79306	2.502421	1.117869	1	4
sub_sample	79306	1.499344	.5000027	1	2
fod_sub_re~n	79306	1704.793	1046.515	0	8311
hamlet_gro~o	79306	1.354122	.4782494	1	2
second_sta~m	79306	2.211005	.7520984	1	3
hh_id	79306	1.752604	1.041876	1	10
level	79306	1	0	1	1
filler	79306	0	0	0	0
slno_of_in~t	79301	1.954124	5.860412	1	99

```
-more-
```

Comment: From the output above you can see that STATA has calculated summary statistics for all the variables. It can calculate meaningful number only if the variable is numeric. But all numeric variables need not make sense.

- **The first column contains the number of observations for which the summary statistics have been generated.**

  Note: If data is missing for a particular observation, STATA will not include such observations when generating the summary statistics. This is very important to know as it affects the results of your interpretation. Therefore, when reporting summary statistics, ensure that you mention the number of observations that have been included for a particular statistic

- **The second column lists the arithmetic means for each of the continuous variables in the dataset**.
- **The third column lists the standard deviation for each of the continuous variables in the dataset**
- **The fourth and fifth columns show the range of values for each continuous variable.** For example, the variable state_region has a minimum value of 11 and a maximum value of 351.

2. Using the sum Command for a Variable/Variable List: **If you type the sum command followed by a variable or a list of variables, STATA will calculate summary statistics for only the variables you have specified.** The syntax to do this is as follows;

sum <variable list>

Example: Meaningful Variables

summ mpce_30_days mpce_corr

Output: You will see the following output on your STATA screen;

```
. summ mpce_30_days mpce_corr
```

Variable	Obs	Mean	Std. Dev.	Min	Max
mpce_30_days	79306	3130.52	2315.914	17	70826
mpce_corr	79306	678.1063	511.7184	17	29306

Comments: From the above output, you can see that STATA has calculated summary statistics for only those variables listed. We can make the following points from the above output;

- **The summary statistics have been calculated for a total of 79306 observations.**
- **The mean mpce_30_days is 3130.52 (thus it can't be per capita) while the mean correct mpce is 678.11**

3. **Using the sum Command with the Option 'detail':** So far we have seen how to generate the mean, standard deviation and range for variables. **Suppose**

you want to generate the variance, skewness and kurtosis you can do this by using the option 'detail' in conjunction with the command sum. The syntax for this command is as follows;

sum <variable list>, detail.

Note: the placement of the comma in the above command. It is always placed before the word detail. Secondly, within the variable list you can specify one or more variables.

Example:

summ mpce_corr, detail

```
. summ   mpce_corr, detail

                            mpce_corr
-------------------------------------------------------------
      Percentiles      Smallest
 1%     206.4286            17
 5%        279.5            20
10%        325.2          23.8       Obs                79306
25%          417            25       Sum of Wgt.        79306

50%        561.5                     Mean           678.1063
                        Largest       Std. Dev.      511.7184
75%      780.875         20471
90%         1116       23608.67      Variance       261855.7
95%         1425          24363      Skewness       10.47252
99%       2511.6          29306      Kurtosis       336.9046
```

Using the sum Command with Qualifiers ('if' and 'by')

If you want to be specific while generating summary statistics, you can use the qualifiers 'if' and 'by' along with the appropriate logical operators. To illustrate, look at the worked examples below;

Example: What is the distribution of Monthly per capital expenditure of the poor (consistent with our exercise, those whose mpce <=800)

summ mpce_corr if poor==1

```
. su  mpce_corr if poor==1
```

Variable	Obs	Mean	Std. Dev.	Min	Max
mpce_corr	60612	497.8241	149.3487	17	800

Command : ***"by"***

Say I have many categories (social caste), and I want to calculate the mpce for each social caste. This would need many "IF" statements.

Instead:

by social_group: summ mpce_corr

ERROR MESSAGE OFTEN:

```
. by social_group: summ mpce_corr
not sorted
```

Before using "by" one has to sort the data by social_group

Therefore

sort social_group

by social group: summ mpce_corr

```
. by social_group: summ mpce_corr

-> social_group = scheduled tribe
    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
   mpce_corr |      12694    641.9039    410.8816        20   7456.667

-> social_group = scheduled caste
    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
   mpce_corr |      13929    557.2948    317.0127        17    6907.25

-> social_group = other backward class
    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
   mpce_corr |      30116     651.757    487.6949        25      29306

-> social_group = others
    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
   mpce_corr |      22502    808.5258    647.1859       23.8      24363

-> social_group = .
    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
   mpce_corr |         65     696.089    450.3869     220.2       3204
```

A Short cut way is

bysort social_group: summ mpce_corr

which gives the same output as above...

Now possibilities immense...

Lets say you want to generate mean mpce for all social groups and religions

bysort social_group religion: summ mpce_corr

```
. bysort social_group religion: summ mpce_corr
```

-> social_group = scheduled tribe, religion = Hinduism

Variable	Obs	Mean	Std. Dev.	Min	Max
mpce_corr	6845	510.2525	311.8473	20	7456.667

-> social_group = scheduled tribe, religion = Islam

Variable	Obs	Mean	Std. Dev.	Min	Max
mpce_corr	115	941.5246	844.6773	191.5	7154.571

-> social_group = scheduled tribe, religion = Christianity

Variable	Obs	Mean	Std. Dev.	Min	Max
mpce_corr	4203	806.048	442.3482	62.28571	6707.25

-> social_group = scheduled tribe, religion = Sikhism

Variable	Obs	Mean	Std. Dev.	Min	Max
mpce_corr	17	739.7067	335.9103	380.5	1669.25

-> social_group = scheduled tribe, religion = Buddhism

Variable	Obs	Mean	Std. Dev.	Min	Max
mpce_corr	533	803.926	486.8314	119	3550

Not the entire output: Just a snapshot

Simple Tabs (One-way Tabulations)

Simple Tab Command: <u>When the tab command is used with one categorical variable, it reports the frequency of observations within each category as well as their proportion (percentage) of the whole.</u> The syntax for the simple tab command is as follows;

tab <variable name>

Example: For example, if you want to generate a frequency distribution table for the variable 'social_group' in the dataset you can type the following command into the STATA command window and press enter;

tab social_group

Output: You will then see the following output on your STATA screen;

```
. tab social_group
```

social group	Freq.	Percent	Cum.
scheduled tribe	12,694	16.02	16.02
scheduled caste	13,929	17.58	33.60
other backward class	30,116	38.01	71.60
others	22,502	28.40	100.00
Total	79,241	100.00	

If you want to also see missing...

tab social_group, missing

```
. tab social_group, missing
```

social group	Freq.	Percent	Cum.
scheduled tribe	12,694	16.01	16.01
scheduled caste	13,929	17.56	33.57
other backward class	30,116	37.97	71.54
others	22,502	28.37	99.92
.	65	0.08	100.00
Total	79,306	100.00	

' tab can be combined with 'if' and 'by'

Example: Social Group of the poor

```
. tab social_group if poor==1, missing
```

social group	Freq.	Percent	Cum.
scheduled tribe	9,864	16.27	16.27
scheduled caste	12,116	19.99	36.26
other backward class	23,856	39.36	75.62
others	14,723	24.29	99.91
.	53	0.09	100.00
Total	60,612	100.00	

Example: In each religion: Social Group of the poor

Snapshot (Not complete Output)

```
. bysort religion:  tab social_group if poor==1, missing
```

```
-> religion = Hinduism
```

social group	Freq.	Percent	Cum.
scheduled tribe	6,145	12.86	12.86
scheduled caste	11,122	23.28	36.14
other backward class	20,862	43.67	79.81
others	9,615	20.13	99.94
.	29	0.06	100.00
Total	47,773	100.00	

```
-> religion = Islam
```

social group	Freq.	Percent	Cum.
scheduled tribe	69	1.00	1.00
scheduled caste	83	1.20	2.20
other backward class	2,349	33.93	36.13
others	4,416	63.79	99.91
.	6	0.09	100.00
Total	6,923	100.00	

Cross Tabs (Two-way tabulations)

1. Cross Tabs: When the tab command is used with two categorical variables it creates a two-way frequency table or a cross-tabulation of the variables.

Therefore, cross tabs are useful when examining the relationship between the 2 variables. The syntax for the simple cross-tab command is as follows;

tab <variable name1> <variable name2>

Note: <u>The first variable in your tab command will be displayed in the rows of the table, and the second variable will be displayed in the columns. Changing their order will change the interpretation of the table.</u>

Example:

tab social_group poor

```
. tab social_group poor
```

social group	poor 0	1	Total
scheduled tribe	2,830	9,864	12,694
scheduled caste	1,813	12,116	13,929
other backward class	6,260	23,856	30,116
others	7,779	14,723	22,502
Total	18,682	60,559	79,241

More meaningful to have proportions

tab social_group poor, row

row: This calculates row proportion. Therefore in the above table, it would calculate the proportion of schedule tribe who are poor

social group	poor 0	1	Total
scheduled tribe	2,830 22.29	9,864 77.71	12,694 100.00
scheduled caste	1,813 13.02	12,116 86.98	13,929 100.00
other backward class	6,260 20.79	23,856 79.21	30,116 100.00
others	7,779 34.57	14,723 65.43	22,502 100.00
Total	18,682 23.58	60,559 76.42	79,241 100.00

Another cut: How much of the poor and non poor are in each social category

tab social_group poor,col

col: calculates the column percentage

```
. tab social_group poor, col nokey
```

social group	poor 0	1	Total
scheduled tribe	2,830 15.15	9,864 16.29	12,694 16.02
scheduled caste	1,813 9.70	12,116 20.01	13,929 17.58
other backward class	6,260 33.51	23,856 39.39	30,116 38.01
others	7,779 41.64	14,723 24.31	22,502 28.40
Total	18,682 100.00	60,559 100.00	79,241 100.00

- You can do row and col together
- You can combine cross tabs with "bysort"
- You can combine cross tabs with "If"

- The additional option ***nofreq*** suppresses the frequency numbers and reports percentages depending on row and col option

Multipliers

If you are interested in Standard Errors (not reported usually in NSS tables): then you have to declare the data set as a survey data set and specify strata and second stage strata (ADVANCED MATERIAL: look at svyset command. After declaring the data set with the appropriate strata, the same commands as above go through and standard errors are adjusted for stratification automatically)

If you are interested in Means and Proportions. you can look at the multipliers as pure frequency weights.

STATA does not like non integer frequency weights. So first step is a rounding off

gen freq=round(weight)

These are household multipliers. For Individual multipliers, one has to mutliply by household size.

Thereafter the same commands as above go through except syntax becomes

summ <variable name> [fw=freq]

tab < variable name> [fw=freq]

All If, bysort cross tab commands still work

Abhiroop Mukhopadhyay, Assistant Professor, Indian Statistical Institute (Delhi): *abhiroop@isid.ac.in*

Merging Files

To merge two files, call the bigger file "master", and the smaller "user". We need to merge the two files by a common identifier. Let that be called common_id (unique household id or individual id).

Steps

1. Open the "user.dta" file and sort by common_id and resave it

2. Open the "master.dta" file and sort it by common_id

then

merge common_id using user.dta

A variable _merge is automatically created

It takes the value 1 if the common_id is in the "master.dta" file but not in the "user.dta" file. It takes the value 2 if the _id is in the "user.dta" file but not in the "master.dta" file. It takes the value 3 if its in both files.

Merging can take place on basis of two variables also...

If common_id is the unique household number and personal serial number (person_srl_no) is unique within the household (1,2,3 ...) then both together define the unique person identification number

In that case merging at the individual level can be on basis of

merge common_id person_srl_no using user.dta

Excel Portability of Output on Results Window

Every table can be copied and pasted directly from the result window onto excel. Copy as Table option retains the formatting. There are also special free programs for generating even nicer formatted tables.

Abhiroop Mukhopadhyay, Assistant Professor, Indian Statistical Institute (Delhi): *abhiroop@isid.ac.in*

ASSIGNMENT

Three files are given to you.

level02.dta: Contains household characteristics

level04.dta: Contains Household roster level information: age and gender of members of the household

level05.dta: Contains Individual principal status information

Exercise 1: By dropping or keeping, I want three data sets that keep the following variables:

level02.dta : hh_id, rel, social_grp, land_own, nregs_job_card

level04.dta : p_id hh_id weight sector state_region district sub_round sex age

level05.dta: prin_act hh_id p_id weight state state_str

After dropping variables, they should be saved as level02new.dta. level04new.dta and level05.dta

Exercise 2: Describe the variables in these data set

Exercise 3: Merge the three data sets, two at a time.

1. First merge level04new and level05 new on the basis of p_id (if you want, you can save it and call it level45.dta)
2. Then merge this file with level02new on the basis of hh_id

Exercise 4:

A person is defined as in the labour force if the principal activity numbers are between 11 and 81. Based on this, calculate the labour force participation rate for

1. All India all persons/ All India men and All India women
2. Rural persons/ Rural men and Rural Women
3. Urban persons/Urban men and Urban Women

Abhiroop Mukhopadhyay, Assistant Professor, Indian Statistical Institute (Delhi): *abhiroop@isid.ac.in*