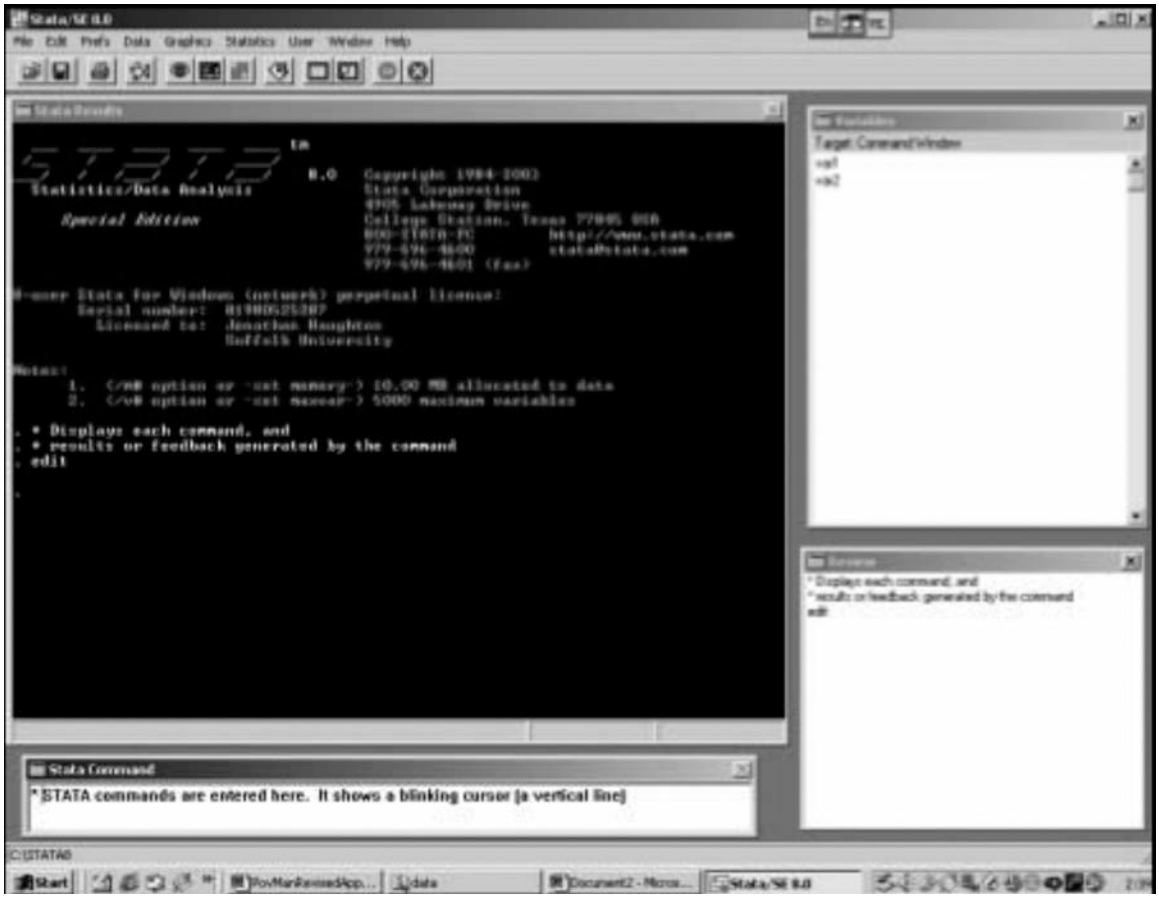
STATA Preliminary

STATA is a statistical software package that offers a large number of statistical and econometric estimation procedures. With STATA we can easily manage data and apply standard statistical and econometric methods such as regression analysis and limited dependent variable analysis to cross-sectional or longitudinal data. STATA is widely used by analysts working with household survey data.

1. Getting Started

1.1 Starting STATA

Start a STATA session by double-clicking on the STATA icon in your desktop. The STATA computing environment comprises four main windows. The size and shape of these windows may be moved about on the screen. Their general look and description are shown below:



It is useful to have the **Stata Results** window be the largest so you can see a lot of information about your commands and output on the screen. In addition to these windows STATA environment has a menu and a toolbar at the top (to perform STATA operations) and a directory status bar at the bottom (that shows the current directory). You can use menu and toolbar to issue different STATA commands (like opening and saving data files), although most of the time it is more convenient to use the **Stata Command** window to perform those tasks.

1.2 Opening a Dataset

You open a STATA dataset by entering following command in the **Stata Command** window:

```
set mem 10m
cd c:\intropov\data
use ind
```

set mem allows you to set the amount of memory STATA will take from your computer (in our case, 10 Mb). **cd** stands for "change directory": from here further, STATA will assume that all the files you are working on are located in the directory **c:\intropov\data**. Change the directory **c:\intropov\data** with the directory where your files are located. Note that you can also open a file by typing each time the directory where it is located (in our case, **use c:\intropov\data\ind**) but this will make your job more complicated.

Please note the following points:

⇒ STATA assumes the file to be in STATA format with an extension **.dta**. So, typing **ind** is the same as typing **ind.dta**.

⇒ We can only have one data set open at a time in STATA. So if we open another data set **hh.dta**, we will be replacing **ind.dta** with **hh.dta**. Fortunately, STATA does not allow you do to this without warning, unless you type something like **use hh.dta, clear**.

1.3 Saving a Dataset

If you make changes in an open STATA data file and want to save those changes, you can do that by using the STATA **save** command. For example, the following command saves the **ind.dta** file:

```
save ind.dta, replace
```

You can optionally omit the filename here (just **save, replace** is good enough). The **replace** option unambiguously tells STATA to overwrite the pre-existing original version

with the new version. If you do NOT want to lose the original version, you have to specify a different filename in the **save** command.

1.4 Notes on STATA Commands

Here are some general comments about STATA commands:

⇒ STATA commands are typed in lower case.

⇒ All names, including commands or variable names, can be abbreviated as long as there is no ambiguity. So **describe, des** or simply **d** do the same job as there is no confusion.

⇒ In addition to typing, some keystrokes can be used to represent a few STATA commands or sequences. The most important of them are the **Page-Up** and **Page-Down** keys. To display the previous command in the **Stata Command** window, you can press the **Page-Up** key. You can keep doing that until the first command of the session appears. Similarly, the **Page-Down** key displays the command that follows the currently displayed command in the **Stata Command** window.

⇒ Clicking once on a command in the **Review** window will put it into the **Stata Command** window; double clicking it will tell STATA to execute the command. This can be useful when commands need to be repeated, or edited slightly in the **Stata Command** window.

2. Working with data files: looking at the content

From now on, we will mostly list the command(s) and not the results, to save space. To go through this exercise open the **hh.dta** file, as we will use examples extensively from this data file.

2.1 Listing the variables

To see all variables in the data set, use the **describe** command (fully or abbreviated). This command provides information about the data set (name, size, number of observations) and lists all variables (name, storage format, display format, label). To see just one variable or list of variables use the command followed by the variable name(s).

EXERCISE 1 - Answer the following questions using ind.dta:

- How many variables in the file ind.dta? _____
- Write their names: _____

Now open the file hh.dta:

- How many variables in the file hh.dta? _____
- Write their names: _____

2.2. Summarizing data

The very useful command **summarize** (which may be abbreviated as **sum**) calculates and displays a few of summary statistics, including means and standard deviations. If no variable is specified, summary statistics are calculated for all variables in the data set. The following command summarizes the household size and education of the household head:

```
sum age sex
```

Any observation that has a missing value for the variable(s) being summarized is excluded from this calculation by STATA (missing values are discussed later). If we want to know the median and percentiles of a variable, we need to add the **detail** option (abbreviated d):

```
sum age sex, d
```

A great strength of STATA is that it allows for the use of weights. The **weight** option is useful if the sampling probability of one observation is different from another. In most household surveys, the sampling frame is stratified, where the first primary sampling units (often villages) are sampled, and conditional on the selection of primary sampling unit, secondary sampling units (often households) are drawn. Household surveys generally provide weights to correct for the sampling design differences and sometimes data collection problems. The implementation in STATA is straightforward:

```
sum age sex [aw=weight]
```

Here the variable **weight** has the information on the weight to be given to each observation and **aw** is a STATA option to incorporate the weight into the calculation.

For variables that are strings, **summarize** will not be able to give any descriptive statistics except that the number of observations is zero. Also, for variables that are categorical (e.g. illiterate = 1, primary education = 2, higher education = 3), it can be difficult to interpret the output of the **summarize** command. In both cases, a full tabulation may be more meaningful, which we will discuss next.

Many times we want to see summary statistics by group of certain variables, not just for the whole dataset. Suppose we want to see mean age, and sex, by region. We could use a

condition in the sum command (for example, **sum age sex if region == 1**), and so on for the other regions, but this is not convenient if the number of categories in the group is large.

There is a simpler solution. First, sort the data by the group variable (in this case, region). You can check this by issuing **describe** command after opening each file. The **describe** command, after listing all the variables, indicates whether the data set is sorted by any variable(s). If there is no sorting information listed or the data set is sorted by a variable that is different from what you want it to be, you can use the **sort** command and then save the data set in this form. The following commands sort the data set by **region** and show summary statistics of family size and education of household head by region:

```
sort region
by region: sum age sex
```

EXERCISE 2 - Answer the following questions using ind.dta:

- Write the mean of the variables capturing age (_____) and sex (_____)
- Suppose that sex = 1 represents a male; what's the percentage of males in the sample? _____

Now summarize age by sex:

- Male (sex = 1): mean of age: _____
- Female (sex = 0): mean of age: _____

2.3 Frequency distributions (tabulations)

We often need frequency distributions and cross tabulations. We use the tabulate (abbreviated **tab**) command to do this. The following command gives the regional distribution of the households:

```
tab region
```

The following command gives the gender distribution of household heads in region 1:

```
tab sex if region==1
```

In passing, note the use of the **= =** sign here. It indicates that if the regional variable is identically equal to 1, then do the tabulation.

We can use the **tabulate** command to show a two-way distribution. For example, we might want to check whether there is any gender bias in the education of individuals. We

use the following command:

```
tab educ sex
```

To see percentages by row or columns we can add options to the **tabulate** command:

```
tab region sex, col row
```

EXERCISE 3 - Answer the following questions using ind.dta:

- In region = 1, what is the percentage of females? _____ And in region = 2? _____
- In which region women are more than men, in relative terms (i.e. in terms of percentage)? _____
- Consider the whole sample; what is the percentage of people living in region = 1? _____ And in region = 2? _____ And in region = 3? _____ And in region = 4? _____
- Now consider only women; what is the percentage of women living in region = 1? _____ And in region = 2? _____ And in region = 3? _____ And in region = 4? _____

2.4 Distributions of descriptive statistics (table command)

Another very convenient command is **table**, which combines features of the **sum** and **tab** commands. In addition, it displays the results in a more presentable form. The following **table** command shows the mean of family size, and education of household head, by region:

```
table region, c(mean age mean educ)
```

2.5 Missing Values in STATA

In STATA, a missing value is represented by a dot (**.**). A missing value is considered larger than any number. The **summarize** command ignores the observations with missing values and the command does the same, unless forced to include missing values.

2.6 Counting observations

We use the **count** command to count the number of observations in the data set. The **count** command can be used with conditions. The following command gives the number of individuals older than 50:

```
count if age>50
```

EXERCISE 4 - Answer the following questions using ind.dta:

- How many observations in the file? _____
- Now count people aged 18 and older. How many observations in the sample? _____
- Now count people living in region = 1. How many observations in the sample? _____

3. Working with data files: changing data set

3.1 Generating new variables

In STATA the command **generate** (abbreviated **gen**) creates new variables, while the command **replace** changes the values of an existing variable. The following commands create a new variable called **old**, then set its value to 1 if an individual is 18 or older and to 0 otherwise:

```
gen old=1 if age>=18
replace old=0 if age<18
```

STATA provides many useful functions to be used in **generate** and **replace** commands, for example **mean(.)** or **max(.)**. For example, in the **ind.dta** file. the following command calculates the maximum share of employment among four sectors for each household:

```
gen maxhr=max(snaghr,saghr,wnaghr,waghr)
```

An extension of the **generate** command is **egen**. Like the gen command, the **egen** command can create variables to store descriptive statistics like the mean, sum, maximum and minimum or other statistics. For example, an alternative way to create the **maxhr** variable is:

```
egen maxhr=rmax(snaghr saghr wnaghr waghr)
```

Note the difference in syntax. The more powerful feature of the **egen** command is its ability to create statistics involving multiple observations. For example, the following command creates average individual employment hours:

```
egen avgemphr=mean(iemphr)
```

All observations in the data set get the same value for **avgemphr**.

The following command creates a variable, called **mean_age**, which captures the mean of the age of individuals within each household :

```
sort hhcode
egen mean_age=mean(age), by(hhcode)
```

Finally, a (non-exhaustive) list of the most common operators used in STATA:

Arithmetic		Relational		Logical	
^	power	>	greater than	!	not
*	multiplication	<	less than	~	not
/	division	>=	> or equal	|	or
+	addition	<=	< or equal	&	and
-	subtraction	==	equal		
		!=	not equal		
		~=	not equal		

EXERCISE 5 - Answer the following questions using ind.dta:

- Create a variable called new_var; new_var = 1 if the individual is female and aged 18 and older; 0 otherwise:
  - How many individuals are females aged 18 and more? _____ And in percentage terms? _____
- Now we want to create a variable capturing the mean of the age of schooling in each household; for example, if the household is composed by 3 individuals, whose age of schooling are respectively 0, 3 and 6 years, the variable mean_school = (0 + 3 + 6)/3 = 3. First of all, make sure that the sample is sorted by household code; then, by using the command egen and the right operator, create a new variable called mean_school; summarize mean_school:
  - What is the mean of this new variable? _____ And its standard deviation? _____

3.2 Renaming variables

You can change a name of a variable by using the following command:

```
rename mean_school mean_edu
```

The variable **mean_school** is now called **mean_edu**.

3.3 Labeling variables

You can attach labels to variables to give a description to them. For example, the variable **mean_edu** does not have any label now. You can attach a label to this variable by typing:

```
label variable mean_edu "mean of the years of schooling"
```

In the **label** command, **variable** can be shortened to **var**. Now to see the new label, type:

```
describe mean_edu
```

3.4 Keeping and Dropping Variables and Observations

We can select variables and observations of a data set by using the **keep** or **drop** commands. Suppose we have a data set with 6 variables: **var1, var2, … , var6**. We would like to keep a file with only three of them, say **var1, var2,** and **var3**. You can use either of the following two commands:

```
keep var1 var2 var3
drop var4 var5 var6
```

We can also use relational or logical operators. For example, the following command drops those observations where the head of the household is 80 or older :

```
drop if age>=80
```

Or:

```
keep if age<80
```

The above two commands give the same result.

3.5 Merging data sets

STATA can only have one data set in memory at a time. However, on many occasions one needs variables that are spread over two or more files and would like to combine those files for the purpose of analysis. For example, we want to see how individual's education varies by the gender of the head of the household. Since the gender variable (**sexhead**) and the individual's education (**educ**) come from two different files (**hh.dta** and **ind.dta**) we have to merge these two files to do the analysis. We want to combine these two files at the household level, so the variable that is used for merging is **hhcode** (this is the merge variable). Before merging is done, both files must be sorted by the merge variable. The following command opens, sorts and saves the **ind.dta** file:

```
use ind,clear
sort hhcode
save, replace
```

Once both data sets have been sorted, they can be merged, as follows:

```
use hh, clear
sort hhcode
merge hhcode using ind
drop _merge
```

In the context **hh.dta** is called the master file (this is the one that remains in the memory before merging) and **ind.dta** is called the using file.

EXERCISE 6 - Answer the following questions using ind.dta and hh.dta:

- Merge ind.dta using hh.dta and calculate the household size: how many households are composed by exactly 10 individuals? _____
- Create a new variable, called income_pc, capturing income per capita. Keep only the variables called hhcode and income_pc; enter the following code:

  ```
  collapse income_pc, by(hhcode)
  ```

- Suppose that the poverty line is 10 unities per days per capita. What is the percentage of families **below** the poverty line? _____