# Advanced Methods in Impact Assessment Workshop

## Day 1: The Creation of Knowledge through Impact Assessments

This is the first of our data exercises and will give you a chance to work with real data and apply the techniques you learned during the morning lecture sessions. For most of the days we will be using the Village Dynamics of South Asia (VDSA) panel data set collected by ICRISAT. We will utilize the recent high frequency parcel level production data from households in India. For the data exercises concerning RCTs, we will use data from a real RCT on the effects of marketing in encouraging households to purchase index insurance. This RCT was conducted in conjunction with ICRISAT, again in India.

To get started, we will take the "raw" VDSA data and prepare it for analysis. This process will be useful for two reasons. First, it will provide you with a chance to familiarize (or re-familiarize) yourself with Stata. Second, since we will be using this data throughout the workshop, these initial exercises will get the data "regression ready" so that we will not need to spend time on Day 3 or Day 4 preparing a data set.

There are three objectives for today's exercises:

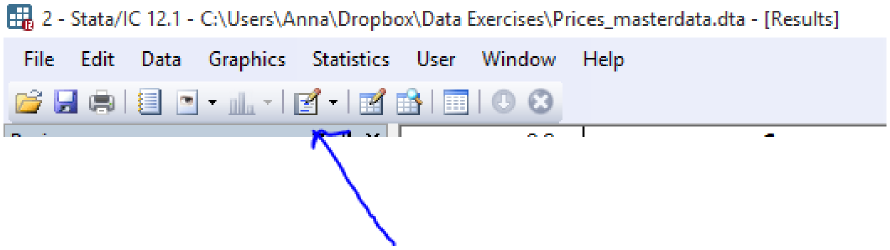
1. Become familiar with Stata and the data we will analyze throughout the workshop.
2. Preparing data for analysis on subsequent days.
3. Use real data to illustrate the role of confounding factors in impact assessment.

## Introduction to Stata

To get started, load into Stata the data set `India_Prod_Data.dta`, shared via dropbox. These data are parcel-level production data for the years 2010 and 2011. The data set contains parcel-level observations on qualitative inputs and outputs for a number of different crops. It also contains observations on household-level characteristics and a few observations of village-level characteristics. As you move through the data set can you identify which observations come from the parcel level, household level, and village level?

Before we begin to modify or analyze the data at all we want to open a `.log` file and create a `.do` file. It is good practice to always record how you manipulate or clean data so that others can replicate your work. Or so that you can reconstruct your work in case you find later that you have made a mistake.

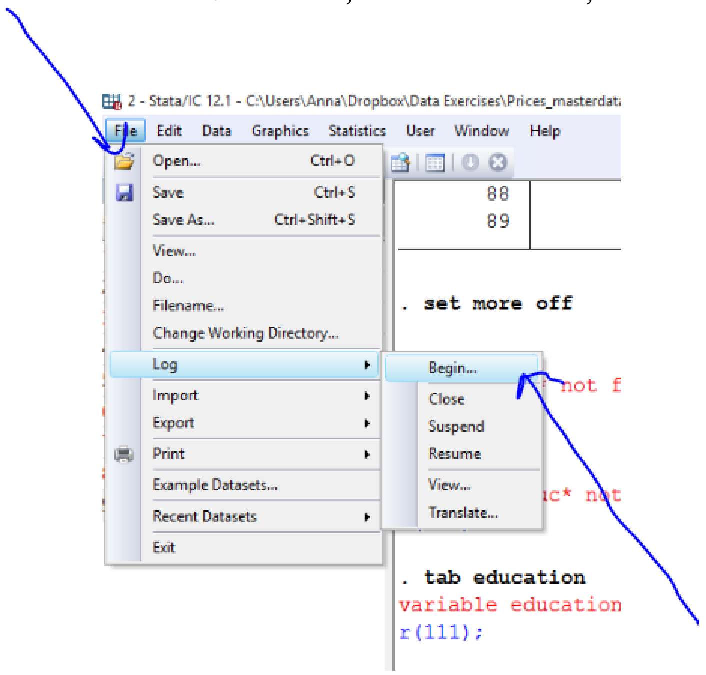
So first, create a `.do` file. Using a `.do` file will help track what you have done, if you want to return to your work later. These are the Stata equivalent of macros. To do create a `.do` file, click on this button.


2 - Stata/IC 12.1 - C:\Users\Anna\Dropbox\Data Exercises\Prices_masterdata.dta - [Results]
File
Edit
Data
Graphics
Statistics
User
Window
Help

Instead of entering the following commands in the command window in Stata, type them in the `.do` file that we opened. That will allow you to return to your work for reference at a later date. You may want to start with the command `set more off`, otherwise when you run Stata, you will have to continually press the space bar in order for Stata to scroll down the screen when the output fills the results table.

Another way to create a `.do` file is to type commands into the command window. When Stata executes a command, it also copies the line to the Review window. While hovering over the Review window, you can press the control key, and you can select all the command lines and copy them into the `.do` file editor. You will have to edit out the commands that had errors. One way to do that is to click “rc” on the top right corner of the Review window. Then you can copy and paste only the lines without errors.

Now, create a `.log` file. Using a log-file will create a `.log` file which can later be translated to `.pdf`. This can help you track commands and results. To do this, click on this button, under the file menu



```
      88
      89

. set more off

       not f

     ic* not

. tab education
variable education
r(111);
```

Alternatively, you can type `log using <<filename.log>>` where you replace `filename` with whatever you want to call your file. So, start up a `.log` file and call it `dataprep.log`. Remember you will need to close the `.log` file at the end either by using the dropdown menu or by typing `log close`.

When you’re done with your `.log` file, you can translate it to a `.pdf` file by using the dropdown menu and selecting translate. Alternatively you can type, with the appropriate location identified:

```
translate "C:\Users\Research\DataExercises\samplelog.smcl"
     "C:\Users\Research\Data Exercises\samplelog.pdf"
```

Now that you have started a log of your work, let’s begin to look at the data set. To see all the variables in a data set, use the `describe` command. This command provides information about the data set, including the name, size, and number of observations. It also lists all variables, including name, storage format, display format, and label. To see just one of a smaller list of the variables, use `describe`, followed by the variable name or names.

The `summarize` command displays a few summary statistics, including means and standard deviations. If no variable is specified, summary statistics are calculated for all variables in the data set.

If interested in looking at summary statistics by a group of certain variables, not for the entire data set, it is necessary to “sort” the data. Sort the data by the group variables of interest. Then, it is possible to consider your variable of interest by the other, for example:

```
sort crop_name
by crop_name: sum output lab_q fert_q irr_q mech_v pest_v
```

There are three additional ways to generate summary statistics:

- `tabstat` allows you to list the statistics for a variable of interest that you want to consider in a single table. It is also possible to condition on another variable.
- `tabulate` allows for consideration of frequency distributions and cross-tabulations.
- `table` combines the features of the `sum` and `tab` commands. It also gives a more presentable form of the statistics.

Using the same variables we used to summarize the data, examine the data using `tabstat, tab,` and `table`. For variables with many values, such as `output`, the tables Stata generates will be very large. So you may want to experiment with `tab` and `table` with variables that do not have many values.

There are three ways that can be useful for considering / learning more about your data:

- `count` is used to count the number of observations in a data set. It is also possible to condition on other variables.
- `distinct` tells you the number of observations and the total number of distinct observations for a variable of interest.
- `duplicates` highlights potential duplicate observations in the data. You can report, tag, or drop duplicate observations (using commands with the same name).

To use the `distinct` command you will most likely need to install the `distinct` package. Stata has many additional commands written by third-parties that require installation. To install `distinct` (or any other external package) type `ssc install distinct`. This command will download and install the package.

Now that we have `distinct` installed, let's use the command. Simply type `distinct prcl_id`. How many observations are in the data and how many distinct parcels are there in the data? How many distinct households and villages?

Variables and observations of a data set can be selected using the keep or drop commands. These can also condition these on another variable, for example:

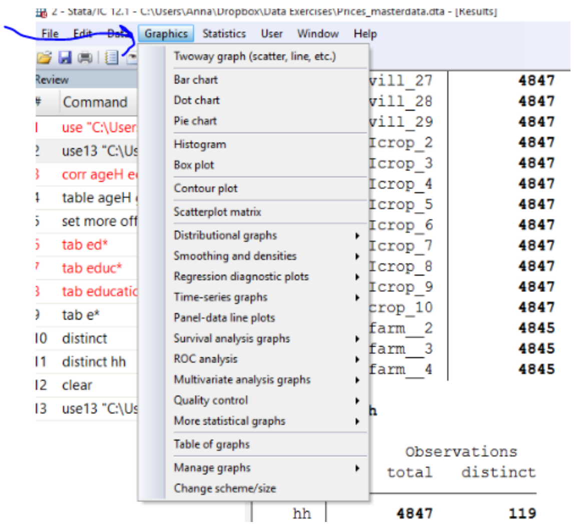
```
keep if var1 > 100
```

So, let's begin to manipulate the data. First, tabulate survey year. Notice that survey years 2009 and 2013 have fewer observations than the other years. This is because the VDSA survey only captured half a year of data in 2009 and has not published the remained of 2013. So, drop all observations that come from 2009 and 2013 so that our data set only contains observations from the years 2010-2012. Two ways to do this are `keep if sur_yr >= 2010 & sur_yr <=2012` or `drop if sur_yr==2009 | sur_yr==2013`. Also, from this point on, we will focus our analyses on wheat. So keep only the observations with wheat (`keep if crop_name=="Wheat"`).

Stata is able to produce a number of basic graphs. As an example, scatterplots can be created using the command:

```
twoway (scatter <<var1>> <<var2>>)
```

To play around with graphs, you can visit the graphics tab, shown below.



```
#   Command
1   use "C:\User
2   use13 "C:\Us
3   corr ageH e
4   table ageH
5   set more off
6   tab ed*
7   tab educ*
8   tab educatic
9   tab e*
10  distinct
11  distinct hh
12  clear
13  use13 "C:\Us
```

```
vill_27 |      4847
vill_28 |      4847
vill_29 |      4847
Icrop_2 |      4847
Icrop_3 |      4847
Icrop_4 |      4847
Icrop_5 |      4847
Icrop_6 |      4847
Icrop_7 |      4847
Icrop_8 |      4847
Icrop_9 |      4847
crop_10 |      4847
farm__2 |      4845
farm__3 |      4845
farm__4 |      4845

h

           Observations
          total   distinct
   hh |    4847        119
```

1. Check the production relationships in the data by creating scatter plots that show the relationship between output and the inputs `plot_area lab_q fert_q irr_q mech_v pest_v`. Create scatter plots for each relationship. What do these plots tell you?

One of the commands you will use most frequently in Stata is the `gen` command, short for `generate`. It allows you to create a new variable. You can either set the value of the new variable to some number (or text string). Or you can set the value of the variable equal to another, existing variable. Or you can set the value equal to some complicated function included numbers and variables. The format is

```
gen <<newvar>> = <<exp>>
```

Where `<<newvar>>` is your new variable name and `<<exp>>` is whatever you want that variable to equal. Once you have created a variable using the `gen` command, any changes must be made with the command `replace`.

2. Create variables that measure each quantitative input in per hectare terms. Then create variables that are logs of each input in per hectare terms.

To label data, variables, or values of variables (for example indicating that for a head of household variable that = `0` is male and = `1` is female), use the command `label`.

```
label var <<var1>> “<<variable label>>”
```

3. Label the variables you just created with labels that include the unit of measure, i.e., (kg/ha) in the case of fertilizer.

Finally we get to running regressions in Stata. Regressions in Stata are in the following form:

```
reg <<dependent variable>> <<independent variable(s)>>
```

Note that Stata requires no punctuation in the regression equation.

4. Run a simple regression with log labor, fertilizer, irrigation, mechanization, and pesticide as independent variables and log yield as the dependent variable. All variables should be logs per hectare. How do you interpret the coefficients?
5. Instead of just taking logs in inputs, create new variables that are the use the Inverse Hyperbolic Sine Transformation. The equation for this transformation is: $\log(x) = \log(x + \sqrt{(x^2 + 1)})$. Call the output variable and each input variable `logyield, loglabor, logfert, logirr, logmech, logpest`. Rerun the regression with these new transformed variables. How have the point estimates changed?

Finally, and this is a good thing to do throughout these exercises, save the transformed data using a name that is descriptive, at least to you.

**The Challenge of Establishing a Causal Effect**

In order to provide a sense of the difficulty in establishing causal effects, we are going to look at our data in two different ways. First, we are going to compare the effect of a hypothetical irrigation intervention on those who received the irrigation treatment versus those who did not. This is our Within/Without comparison. Second, we are going to compare the effect of the irrigation treatment on households before they received the treatment and after they received the treatment. This is our Before/After comparison.

First, generate a new variable called `irr` that equals 1 if the parcel under observation had irrigation greater than zero and equals zero if the parcel had no irrigation (use the `gen` and `replace` commands). This type of variable with values of 0 or 1 is called a binary indicator or dummy variable. Save this data file. We will refer to this data set (with the irrigation indicator, the log transformed variables, without 2009 and 2013, and with wheat production only) throughout the week so make sure you save this version of the data and do not overwrite this file later today or in subsequent days.

*Within/Without Comparison*

Using the transformed data set that you just saved, create a new data set that contains only data from `sur_yr=2011`. Save this data set using a different file name.

6. Generate a table (using the `summarize` command) that shows the average yields for those with the irrigation treatment and those without the irrigation treatment. What do you learn from the table about the impact of the irrigation treatment on yields?
7. Calculate the correlation between the treatment and outcome. Can you determine if the treatment had a positive or negative effect on yields? Can you determine the size of the effect?
8. Do a `ttest` to compare the mean yield by households who received the irrigation treatment with that of the control. What does the test indicate? Is this estimate the intention to treat effect, the effect of treatment on the treated, or the average treatment effect? Explain.
9. Run a regression analysis including only the irrigation treatment variable and log of crop yield as the outcome. What is the result? What is the marginal effect of having irrigation on crop yield? Is this the impact of the irrigation treatment?
10. What control variables might we want to include in a regression to determine the impact of the irrigation project?

Now add the following control variables: `loglabor logfert logmech logpest ageH genderH sizehh logaindex loglindex logtot_acre, logdist`. Our dependent variable should be `logyield`. Note that you will have to use the Inverse Hyperbolic Sine Transformation on the asset index, livestock index, landholding, and distance variables. Also, note that we will refer to this set of 11 variables as our "standard set of control variables" in subsequent exercises. It would be a good idea to save your data at this point.

**11.** After adding the control variables, how do our results change? Try including indicators for `season`. What does the point estimate tell us? Is the coefficient on `irr` the unbiased effect of the treatment? What else could it be?

*Before/After Comparison*

Return to the data set you saved before the Within/Without comparison. Using this data set create a new data set that contains only households that received the irrigation treatment in 2011. Observations will be from 2010 and 2011. To do this, run the following code on the data set you just loaded.

```
drop if sur_yr==2009 | sur_yr==2012 | sur_yr==2013
drop if sur_yr==2011 & irr==0
duplicates tag prcl_id season, generate(dup)
drop if dup==0
```

Before proceeding, check to make sure observations are balanced by typing `tab irr sur_yr`.

**12.** Generate a table (using the `summarize` command) that shows the average yields for households before they received the irrigation treatment and average yields after the irrigation treatment. What do you learn from the table about the impact of the irrigation treatment on yields?

**13.** Do a `ttest` to compare the mean yield by households before and after the irrigation treatment. What does the test indicate? Is this estimate the intention to treat effect, the effect of treatment on the treated, or the average treatment effect?

**14.** Run a regression analysis including only the irrigation treatment variable and the log of crop yield as the outcome. What is the result? What is the marginal effect of having irrigation on crop yield? Is this the impact of the irrigation treatment?

**15.** Now add the control variables. How do our results change? What does the point estimate tell us? Is the coefficient on `irr` the unbiased effect of the treatment? What else could it be?