Advanced Methods in Impact Assessment Workshop

Day 1: The Creation of Knowledge through Impact Assessments

This is the first of our data exercises that 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.

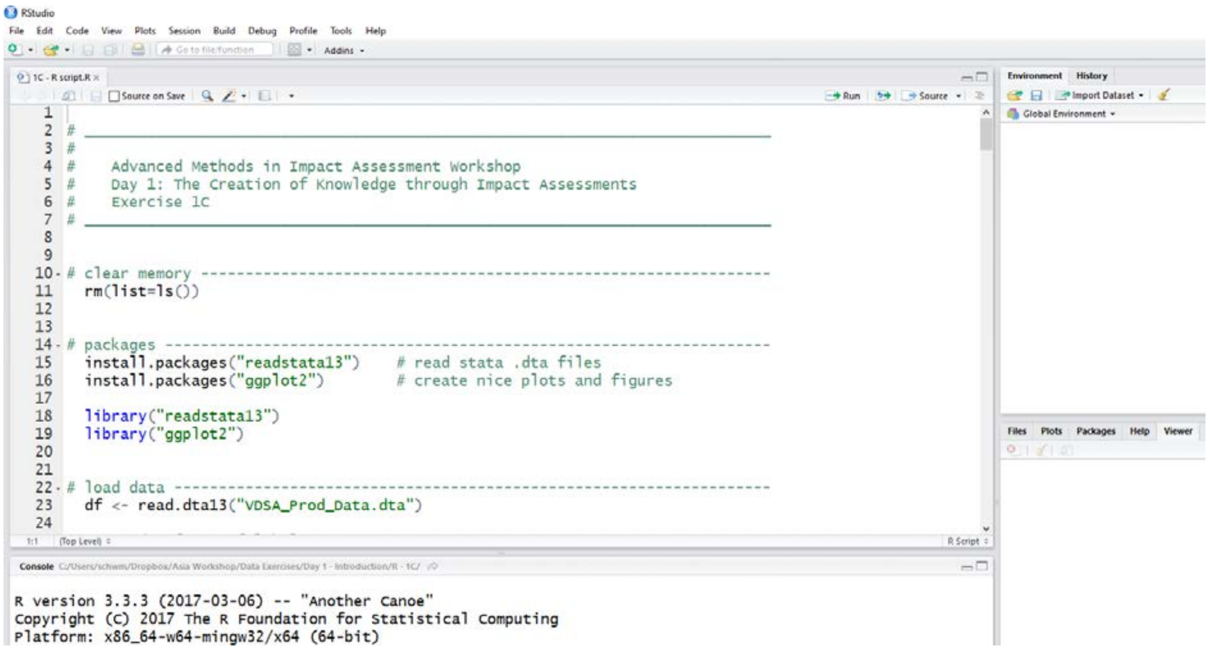
To get started, we will take the "raw" VDSA data and prepare it for analysis. This process will be useful for two reasons. First, it will provide you with a chance to familiarize (or re-familiarize) yourself with **R**. Second, since we will be using this data throughout the workshop, these initial exercises 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 for regressions.

There are three objectives for today's exercises:

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

Introduction to R

To get started, open the file *1C – R script.R* with **Rstudio**. The starting screen should look something like this:


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
1C - R script.R
Source on Save
Run
Source
Environment History
Import Dataset
Global Environment
Files Plots Packages Help Viewer
1:1
(Top Level)
R Script
Console C:/Users/schwm/Dropbox/Asia Workshop/Data Exercises/Day 1 - Introduction/R - 1C/

```

# ______________________________________________________________________
#
#    Advanced Methods in Impact Assessment Workshop
#    Day 1: The Creation of Knowledge through Impact Assessments
#    Exercise 1C
# ______________________________________________________________________


# clear memory ----------------------------------------------------------
rm(list=ls())


# packages --------------------------------------------------------------
install.packages("readstata13")    # read stata .dta files
install.packages("ggplot2")        # create nice plots and figures

library("readstata13")
library("ggplot2")


# load data -------------------------------------------------------------
df <- read.dta13("VDSA_Prod_Data.dta")

```

```
R version 3.3.3 (2017-03-06) -- "Another Canoe"
Copyright (C) 2017 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)
```

In an R script, everything in a line after a # character is read as a comment. Comments are used to explain parts of a script and facilitate documentation. By default, **Rstudio** highlights comments in light green.

The first actual command of the script is the line:

```
rm(list=ls())
```

This command clears the memory before running anything on the script, which is always a good programming practice. To run a specific line in **Rstudio**, highlight the whole line and hit *Ctrl+Enter*

The following command lines install and load the packages required for this exercise:

```
install.packages("readstata13")
install.packages("ggplot2")
library("readstata13")
library("ggplot2")
```

The function `install.packages()` downloads and installs a package from the internet. The `library()` function loads that particular package into the current session.

Packages are a very important part of R. The basic default installation already provides some limited functionality. But the real power of R lies in the large community of independent developers who are constantly implementing new packages to the language.

Now let's read in the data

```
df <- read.dta13("VDSA_Prod_Data.dta")
```

The above command line reads the file `VDSA_Prod_Data.dta` and assigns its information to the object `df`. The pair of characters `<-` is the left assignment indicator of R. It indicates that all the information to the right of it is assigned to the object to its left. There is nothing special about the name `df`, it is just an abbreviation for *data frame*. If you prefer, you can use any other name for it.

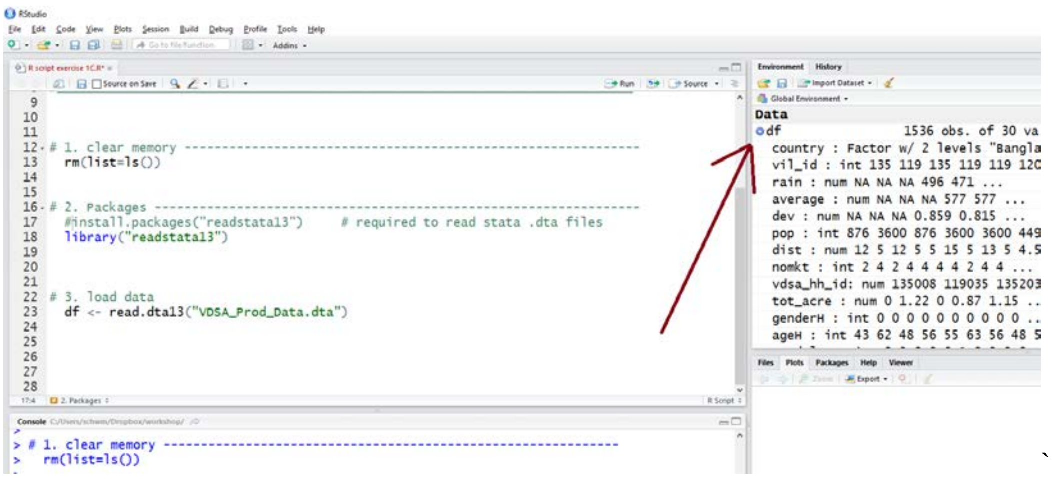
These data are parcel level production data. The data set contains parcel level observations on inputs and outputs for wheat. It also contains observations on household level characteristics and a few observations of village level characteristics. As you move through the data set, can you identify which observations are at the parcel level, household level, and village level?

The file `VDSA_Prod_Data.dta` was created on the specific file format used by **Stata**, so some of its features are not immediately available when loaded in **R**. For example, it is not as easy as in **Stata** to access the variable labels. The following commands extract the variable labels from the original `.dta` file and merge them with the variable names from the original data frame.

```
labels <- as.data.frame(cbind(colnames(df),
                        attributes(df)$var.labels))
colnames(labels) <- c("variable", "label")
```

From the above commands, an additional data frame is created with the variable names and their corresponding labels.

To have a general overview of the variables in your data set, you can take advantage of the "*Data*" pane in the "*Environment*" window of **Rstudio**, which is usually located as the right-upper quadrant of the screen. If you click in blue arrow just to the left of a data frame name (in our case, `df`), it will open a list with all variables within it and some basic information about each variable, such as name, format and the first few observations.



```
# 1. clear memory ------------------------------------------------------------
  rm(list=ls())


# 2. Packages ----------------------------------------------------------------
  #install.packages("readstata13")     # required to read stata .dta files
  library("readstata13")


# 3. load data
  df <- read.dta13("VDSA_Prod_Data.dta")
```

```
Data
df                   1536 obs. of 30 va
  country : Factor w/ 2 levels "Bangla
  vil_id : int 135 119 135 119 119 12C
  rain : num NA NA NA 496 471 ...
  average : num NA NA NA 577 577 ...
  dev : num NA NA NA 0.859 0.815 ...
  pop : int 876 3600 876 3600 3600 449
  dist : num 12 5 12 5 5 15 5 13 5 4.5
  nomkt : int 2 4 2 4 4 4 2 4 4 ...
  vdsa_hh_id: num 135008 119035 135203
  tot_acre : num 0 1.22 0 0.87 1.15 ..
  genderH : int 0 0 0 0 0 0 0 0 0 ..
  ageH : int 43 62 48 56 55 63 56 48 5
```

```
> # 1. clear memory ---------------------------------------------------------
>   rm(list=ls())
```

To view all the data in a data frame, you can use the function `View()`. Note that the "V" in `View` is an uppercase letter. **R** is a case sensitive language, so you must be very careful with the use of uppercase and lowercase in variable names and functions. Therefore, if you run the command `View(df)`, it will open the whole data in the main **Rstudio** window:

The function `summary()` can be used to display the basic summary statistics of a variable, including means, quartiles and number of missing observations. If you ask for the `summary()` of a whole data frame, the summary statistics are calculated for all variables in the data set, for example: `summary(df)`

To see the summary of a particular variable, you can use the summary function indicating the data frame of your variable followed by a `$` character and the name of the variable you want. For example, for the summary of rain, we write: `summary(df$rain).`

Sometimes, you may be interested in looking at summary statistics for a subset of observations, not for the entire data set. There are several ways to achieve this outcome in **R**. One possibility is to use the `by()` function. For example, we can look at the `output` variable by gender of the household head:

```
by(df$output, df$genderH, FUN=summary)
```

The first argument of the function is the object we want to evaluate, in our case the `output` variable. The second argument is the categorical variable used to subset, for which we use the variable `genderH`. Finally, the last argument is the function to be applied for each group, which in our case is the `summary` function.

To summarize a variable by the combination of multiple categorical variables, you can create a temporary concatenated variable as the second argument of the `by()` function using the `paste()` function. For example, to summarize `output` by each combination of `genderH` and `farm_cat` you can write:

```
by(df$output, paste(df$genderH, df$farm_cat), FUN=summary)
```

Another important function for summarizing categorical variables is the `table()` function. For example, to see the distribution of the categorical variable `farm_cat` you can use:

```
table(df$farm_cat)
```

Use the `by()` function to summarize the variable `ageH` by the combination of `genderH` and `farm_cat`. Use `table()` to tabulate the categorical variable `genderH`.

There are other commands that can be useful for learning more about your data:

- `nrow()` can be used to count the number of observations in a data set. It is also possible to group by another variable.
- `duplicated()` indicates duplicated observations in the data. You can use it to drop duplicate observations like the following: `df <- subset(df, !duplicated(df))`
- `unique()` tells you the unique values of a particular variable. to count the number of unique values of a variable, you could use: `length(unique(df$vil_id))`

Use the `length(unique())` command to determine how many unique parcels are in the data. How many distinct households and villages?

Variables and observations of a data set can be selected using the `subset()` command. For example, to keep only observations where some variable `var1 > 100`, you could write:

```
df <- subset(df, var1 > 100)
```

Let's begin to manipulate the data. First, tabulate survey year: `table(df$sur_yr)`. 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 only recently published the remainder of 2013. So, drop all observations that come from 2009 and 2013.

```
df <- subset(df, sur_yr %in% c(2010, 2011, 2012))
```

The characters `%in%` represent the mathematical expression "*belongs to*". The function `c()` creates a vector, so in the above expression it is used to create a vector with elements 2010, 2011 and 2012. Therefore, the complete command above can be read as: `df` receives the subset of `df` where the variable `sur_yr` belongs to the set [2010, 2011, 2012].

R is a very powerful tool for producing visual analysis. A very popular package used to create plots and graphs is `ggplot2`. The syntax may seem complicated at a first glance, but due to its flexibility, it is a tool that is certainly worthwhile learning. For example, to create the histogram of a single variable, e.g. `price`, you can use the command:

```
ggplot(data = df) +
  geom_histogram(aes(x = price), binwidth = 1)
```

The process is similar to create a scatter plot. Instead of `geom_histogram` you can specify a `geom_point` and include an `y` variable to the `aes()`.For example, to create a scatter of `price` and `value` you could write:

```
ggplot(data = df) +
  geom_point(aes(x = price, y = value))
```

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`, `irr_q`, and `pest_v`. Create scatter plots for each of the four relations. What do these plots tell you?

Creating variables in a data frame in **R** is easy. All you need to do is to assign some value to a variable name that does not exist in the data frame. For example, to create a variable called `var1` in data frame *df* and assign the value of 1 to all observations, you could right:

```
Df$var1 <- 1
```

In production economics, we generally want to take log transformations of the data. This allows us to estimate Cobb-Douglas (or Translog) production functions while also allowing us to interpret coefficient estimates as elasticities. However, as you may have noticed from the scatter plot of pesticide, there are a lot of zeros in the data. This creates a problem since log(0) is undefined. One way to deal with the problem of taking the log of a zero is to use the Inverse Hyperbolic Sine Transformation. The equation for this transformation is: $\log(x) = \log(x + \sqrt{(x^2+1)})$. This solves the problem of $\log(0)$ without adding an arbitrating number to the value of the variable.

Now, let's create variables for output and each input in per hectare terms using the Inverse Hyperbolic Sine Transformation. **R** includes a built in function for calculating the Inverse Hyperbolic Sine Transformation, `asinh().` For example, to create a variable called `lny` with the log of `output` divided by `plot_area`, we can run:

```
df$lny <- asinh(df$output/df$plot_area)
```

Now, create similar variables for all inputs and call them `lnl`, `lnf`, `lni`, `lnm`, and `lnp`.

Label the variables you just created with labels that include the unit of measure, i.e., (kg/ha) in the case of fertilizer. The following command lines assign a new label to each new variable and append it to the original `labels` data set.

```
new.vars <- c("var1", "lny", "lnl", "lnf", "lni", "lnm", "lnp")
new.labels <- cbind(new.vars, c("variable with ones",
                "log of output per area (Rs/ha)",
                "log of labor per area (Hr/ha)",
                "log of fertilizer per area (kg/ha)",
                "log of irrigation per area (Lt/ha)",
                "log of pesticid per area (Rs/ha)"))
colnames(new.labels) <- c("variable", "label")
labels <- rbind(labels, new.labels)
```

Finally we can now run regressions in **R**. Basic OLS Regressions in **R** are run in the following form:

```
lm(indep_var ~ dep_var1 + … + dep_varN, data = data_frame_name)
```

Where the first set of arguments are the regression formula and the second indicates the data frame to be used. To visualize a detailed regression output table, you can assign the regression to an object and call the *summary()* of it:

```
r1 <-  lm(indep_var ~ dep_var1 + … , data = df)
summary(r1)
```

2. Run a simple regression with log labor, fertilizer, irrigation, mechanization, and pesticide as independent variables and log yield as the dependent variable. How do you interpret the point estimates on each variable?

Before we move on, let's create a binary indicator for irrigation use called `irr` that equals 1 if the parcel under observation had irrigation greater than zero and equals zero if the parcel had no irrigation.

```
df$irr <- ifelse(df$irr_q > 0, 1, 0)
```

This will be our "treatment" variable. Let's also create log transformed variables of `aindex`, `lindex`, `tot_acre`, and `dist` using the Inverse Hyperbolic Sine. Call them `lnaindex`, `lnlindex`, `lntot_acre`, and `lndist`.

To save this data frame as .csv file called `VDSA_Prod_Data_Ref.csv`, you can use the following command:

```
write.csv(df, "VDSA_Prod_Data_Ref", row.names = FALSE)
```

This will be the file we return to throughout the week.

The Challenge of Establishing a Causal Effect
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.

Within/Without Comparison
Using the transformed data frame that you just saved, create a new data set that contains only data from `sur_yr=2011`. Call it `df_2011`:

3. Generate a table (using the `by()` function) that shows the yields for those with irrigation treatment and those without the irrigation treatment. What do you learn from the table about the impact of the irrigation treatment on the log of yields?
4. Do a `t.test` 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. The code is:

```
t.test(output~irr, data = df_2011)
```

5. Run a regression that includes 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? Why or why not?
6. 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: `lnl`, `lnf`, `lni`, `lnm`, `lnp`, `ageH`, `genderH`, `sizehh`, `aindex`, `lindex`, `tot_acre`, and `dist`. You might want to log transform that variables `aindex`, `lindex`, `tot_acre`, and `dist` but this is not necessary. Our dependent variable should be `lny`. Note that we will refer to this set of 11 variables as our “standard set of control variables” in subsequent exercises.

7. After adding 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?

Before/After Comparison
Return to the data frame used before the Within/Without comparison (`df`). Using this data set create a new data set that contains only households that received the irrigation treatment. To do this, run the following set of commands:

```
# drop observations from 2012
df <- subset(df, sur_yr != 2012)

# count observations per parcel
df$obs <- ave(df$var1, df$prcl_id, FUN=sum)

# identify treated parcels in 2011
df$tr_2011 <- ifelse(df$sur_yr == 2011 & df$irr == 1, 1, 0)

# assign 2011 treatement status to observations in both years
df$tr_2011 <- ave(df$tr_2011, df$prcl_id, FUN=sum)

# subset of observations treated in 2011
df_tr2011 <- subset(df, tr_2011 == 1)

# keep only variables observed in both years
df_tr2011 <- subset(df_tr2011, obs == 2)

# make sure observations are balanced
table(df_tr2011$irr)
```

8. Generate a table (using the `summary` 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?

9. Do a `t.test` 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? Explain.

10. Run a regression that includes 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?

11. Now add the standard set of 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?