# Stata Workshop

## At MINAGRI

Roshni Khincha and Sakina Shibuya
DIME, World Bank

August, 2018



# Section 1

# Why learn stata?

**Excel vs Stata**

Can I use Excel?

## The main reasons to use Stata

- In Excel you make changes directly to the data and save new versions of the data set
- In Stata you make changes to the instructions on how to get from the raw data to the final analysis and save new versions of the instructions
- Since Stata is a more statistics oriented software, processing the data to create analytical products can be a lot easier.

## The main reasons to use Stata

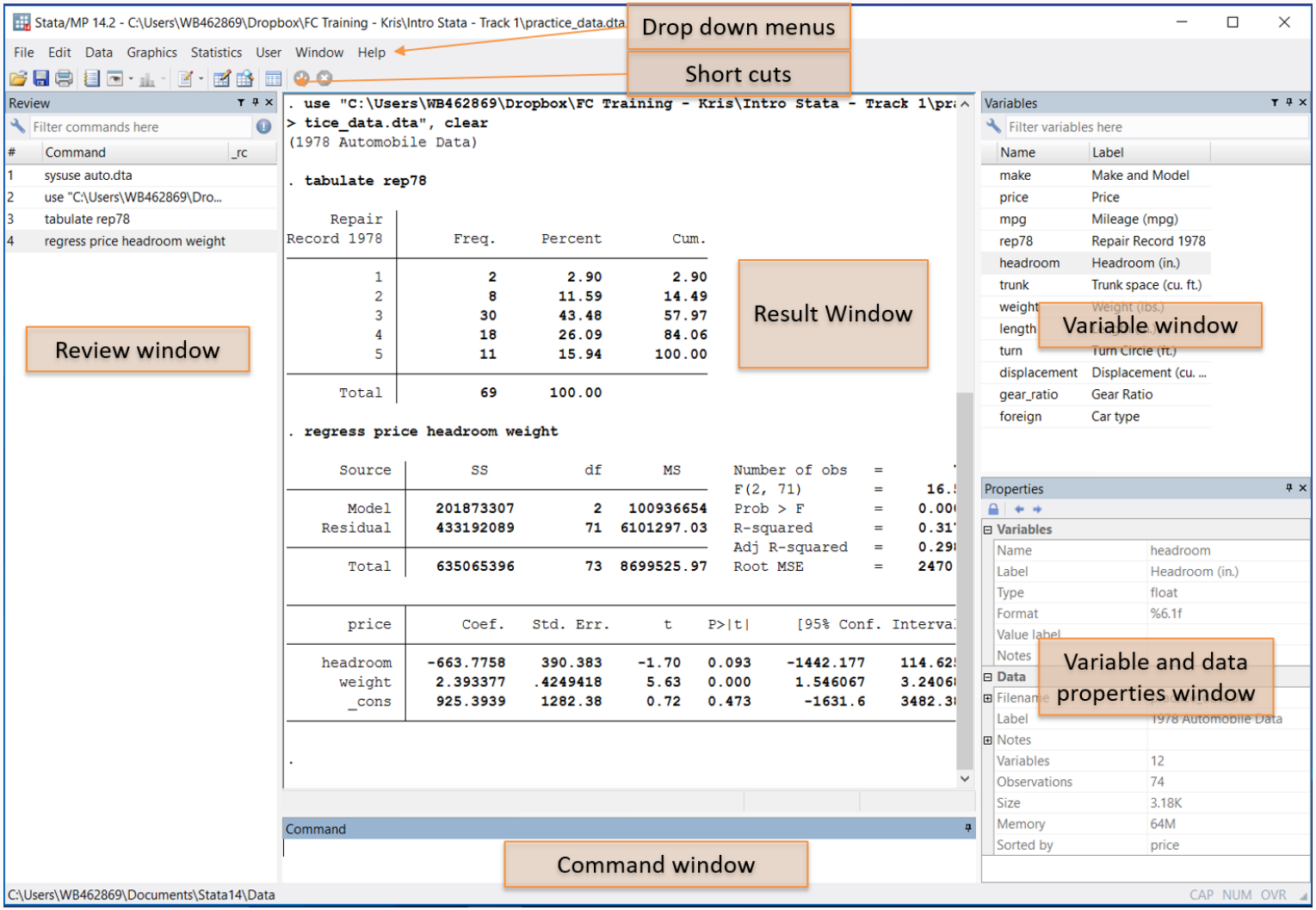
- Powerful tool with may capabilities:
  - Descriptive statistics
  - Inference statistics
  - Complex data analysis
- But its also good for beginner programmers:
  - User friendly interface
  - Relatively easy programming language that can be learned while youre using the software

## What's the fuss about do-files?

- Its through the do-file you communicate your work to other members in your team, both current and future
- Think of the do-files as instructions on how to get from raw data to final report
- For a simple task you can enter commands manually. But for more complex tasks you need to write a recipe, or a list of instructions

## Stata interface

# The Stata interface

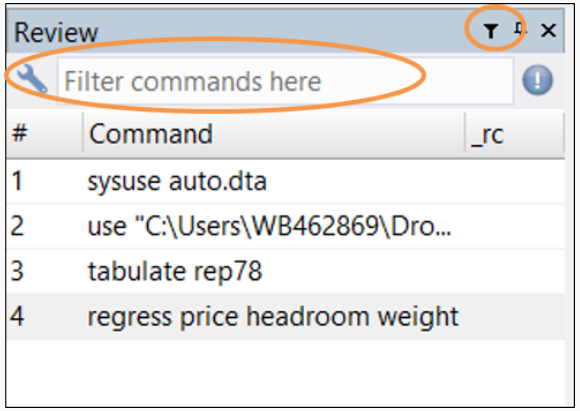


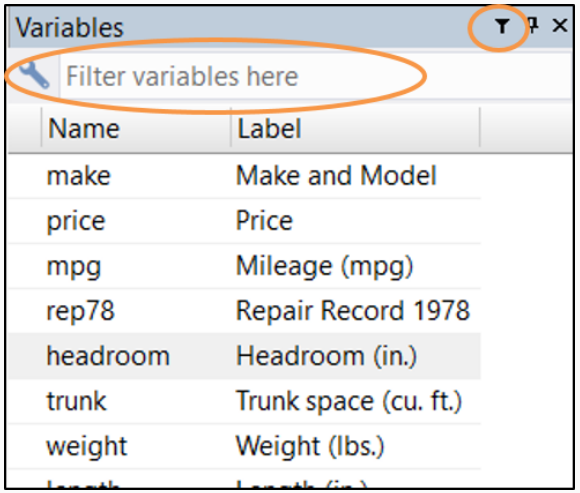
## The review window

- Provides a history of your actions
- A convenient way to bring back your previous commands and modify it to do something new
- Double click on a command you want to use again and it will appear in your command window
  - You can also click in command window and select the commands in the result window by using *PageUp/PageDown* buttons (or *fn+ArrowUp/ fn+ArrowDown* on Mac)
- If a command is red in the review window, it means it did not finish because an error

## Filtering in variable and review windows

- Both the variable and the review window will soon be very crowded. You can then search both of them for commands/variables
- If you do not see the search bar, click the little funnel symbol



## How to open a data set in Stata

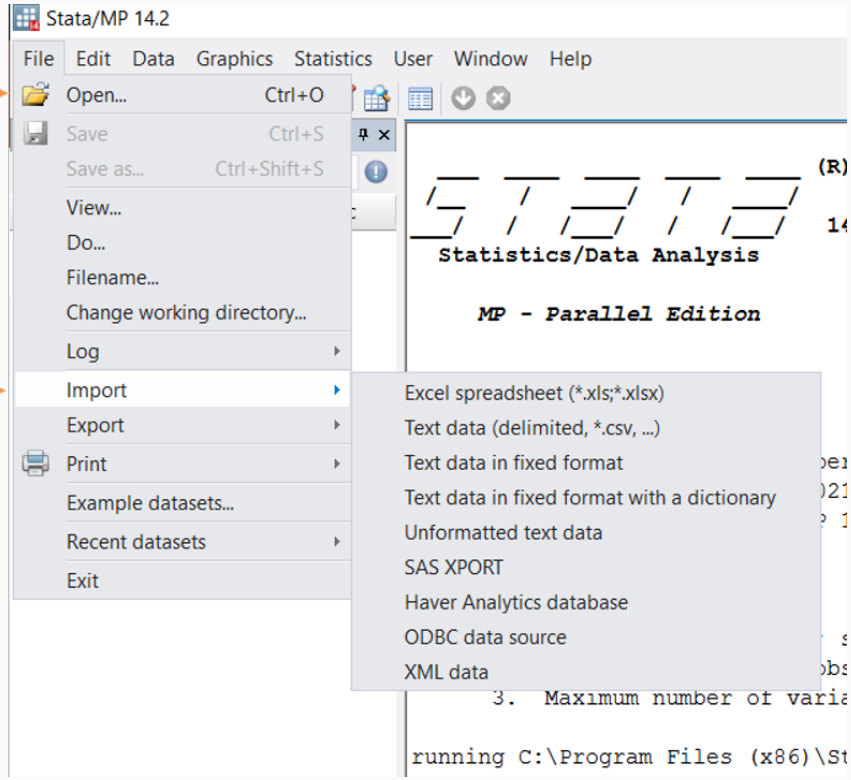
## Three ways to tell Stata what to do

- Drop-down menus
  - An easy place to start but quickly becomes inefficient
- Command window
  - Faster than menus but require that you are familiar with the command
- Do-file
  - The only feasible way to run long instructions
  - Use menus and command window to figure out what you need to write, then copy to a do file

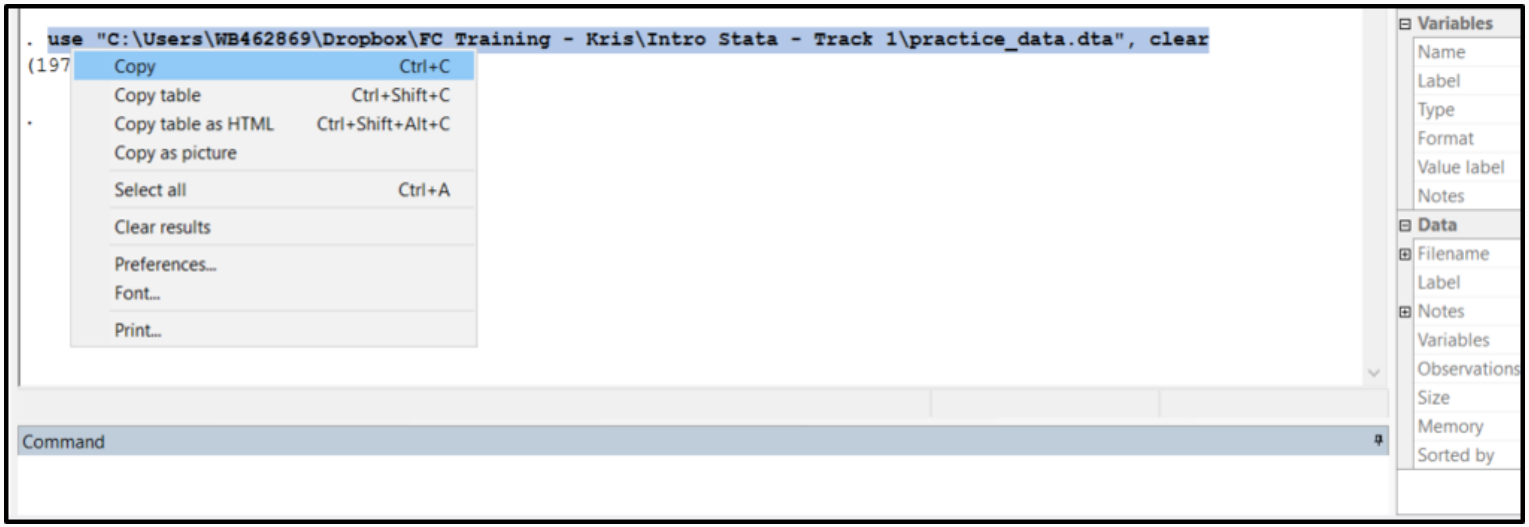
# Open a dataset - menus

Open data sets that are already in Stata format here (.dta)

Open all other types of data sets (.csv, .xslx etc.) here



# Open a dataset - command window




- When you use the menus, Stata produces the code for that action (except for Data Browse)
  - Highlight, right-click and copy the code
  - Paste the code in the command window
  - Hit enter

# Task 1

1. Open Stata and then open the EICV household data set **cs_s0_s5_household.dta** using the menu: File → Open. Navigate to where you saved the material for this lab. Select the data set and click *Open*
2. Browse to check that you have data: Data → Data Editor → Data Editor Browse
3. Describe to get additional information on the data: Data → Describe data → Describe data in memory or in a file.
   - A new window will open
   - Select In memory and press OK

# Task 1

- You can see that one the second command printed information on your screen.
  - The first part is the command used
  - The second part are the results

```
Contains data from C:\Users\WB519128\Dropbox\Work\WB\Mission - Rwanda Feeder Roads\Sta
> ta Training\Data\cs_s0_s5_household.dta
  obs:         14,419
 vars:             76                          28 Jun 2016 09:56
 size:      8,709,076
-------------------------------------------------------------------------------
              storage   display    value
variable name   type    format     label      variable label
-------------------------------------------------------------------------------
hhid            double  %10.0g                Household Identifier_in cross section
province        double  %10.0g     province   Province
district        double  %10.0g     district   District (also stratum CS; panel)
ur2012          double  %10.0g     ur2012     Urban/Rural 2012 (4 categories)
ur2_2012        double  %10.0g     ur2_2012   Urban/Rural 2012 (2 categories)
region          double  %10.0g     region     Region
weight          double  %10.0g                Sampling weight_CS
clust           double  %10.0g                Cluster
rwanda          double  %10.0g     rwanda     All Rwanda
```

## Task 1

- You can perform both tasks by typing the in your command prompt. This will yield the same results
- Type *browse* in the command window and press enter
- Type *describe* and press enter

# Exploring a data set opened for the first time

## Introduction

- To successfully clean a data set you must first understand the data set
- Some terminology:
  - Columns are called variables
  - Rows are called observations

# The EICV data

- For our exercises we will explore part of EICV 4 data
- The data is a household survey collected between 2013 and 2014 by NISR
- It is a cross-section of more than 14 thousand Rwandese households both in rural and urban areas
- Close to 2 thousand of these households form a panel have been also interviewed in EICV 3

# Types of variables

- In Stata, each variable (column) has to be either:
  - string (text) values are red when browsing
  - numeric (number) values are black or blue when browsing
- Numbers **can** be stored as text, but text **cannot** be stored as number
  - Not possible to do computations on numbers stored as text
- Categorical variables should be stored as numeric variables and have labels

# How the data looks

| hhid | province | district | ur2012 | ur2_2012 | region | Weight |
|---|---|---|---|---|---|---|
| 100004 | Kigali Cit | Nyarugenge | Urban | Urban | Kigali Cit | 71.45979 |
| 100005 | Kigali Cit | Nyarugenge | Urban | Urban | Kigali Cit | 71.45979 |
| 100006 | Kigali Cit | Nyarugenge | Urban | Urban | Kigali Cit | 71.45979 |
| 103589 | Southern P | Gisagara | Peri urban | Rural | Rural Sout | 154.7477 |
| 103718 | Southern P | Gisagara | Rural | Rural | Rural Sout | 165.6057 |
| 103719 | Southern P | Gisagara | Rural | Rural | Rural Sout | 165.6057 |
| 103720 | Southern P | Gisagara | Rural | Rural | Rural Sout | 165.6057 |
| 105133 | Southern P | Nyamagabe | Semi urban | Urban | Other Urba | 152.6599 |
| 105134 | Southern P | Nyamagabe | Semi urban | Urban | Other Urba | 152.6599 |
| 105135 | Southern P | Nyamagabe | Semi urban | Urban | Other Urba | 152.6599 |

# How the data actually is

| hhid | province | district | ur2012 | ur2_2012 | region | weight |
|---|---|---|---|---|---|---|
| 100004 | 1 | 11 | 1 | 1 | 1 | 71.45979 |
| 100005 | 1 | 11 | 1 | 1 | 1 | 71.45979 |
| 100006 | 1 | 11 | 1 | 1 | 1 | 71.45979 |
| 103589 | 2 | 22 | 3 | 2 | 3 | 154.7477 |
| 103718 | 2 | 22 | 4 | 2 | 3 | 165.6057 |
| 103719 | 2 | 22 | 4 | 2 | 3 | 165.6057 |
| 103720 | 2 | 22 | 4 | 2 | 3 | 165.6057 |
| 105133 | 2 | 25 | 2 | 1 | 2 | 152.6599 |
| 105134 | 2 | 25 | 2 | 1 | 2 | 152.6599 |
| 105135 | 2 | 25 | 2 | 1 | 2 | 152.6599 |

# Useful commands

- *browse*: see all data in spreadsheet format
- *describe*: list of all variables in memory
  - Total number of variables & observations (size of matrix)
  - Variable name, type, format, value label name, variable label
- *summarize*: Basic statistics for numeric variables
  - Obs (Number of observations), Mean, Std. Dev. (Standard deviation), Min (Minimum), Max (Maximum)
- *tabulate*: frequencies

# More commands

- *codebook*: displays the following for each variable
  - Type (more detail than describe)
  - Number of unique values and number of missing values
  - Range and units
  - Examples of values (strings); tabulations (categorical); or mean, sd and percentiles (continuous)
  - Warnings if embedded blanks (may or may not be ok)
- *labelbook*: displays the following for each stored value label
  - Label definitions
  - Which variables labels are applied to
- *list*: lists all variables and observations
  - Can qualify: *list if price <5000, list in 1/10*
- *summarize, detail* : percentiles, variance, skewness, kurtosis

# Task 2

1. Open the **cs_s0_s5_household.dta** again. Use the command prompt this time.
2. Explore the dataset
   - browse - see the different colors in the columns
   - describe - check the storage type column
   - summarize - are there any statistics that might not make sense to interpret?
3. Learn more about the variable *s5bq3a*, the household estimated rent amount. What values does it take on? What is minimum, maximum, mean of this variable? How many unique values does it have?

```
. use "$data\cs_s0_s5_household.dta", clear
. tabulate s5bq3a
. summarize s5bq3a
. codebook s5bq3a
```

# Task 2

- Learn more about the variable *ur2012*, to learn about the proportion of urban and rural households in Rwanda
  - `. tabulate ur2012`
  - Create now a pie chart: Graphics → Pie chart, select *ur2012* as Category variable and press OK
- Now, create a pie-chart graph for the variable *s5cq7*, the type drinking water source used. This time, use the command prompt!
  - Use the code printed by the previous graph and replace the name of the variable

# Tips and resources

- Using help - Type *help summarize* to get documentation on the summarize function
- Using search - Type *search regression* to get general documentation on running regressions in Stata
- Google - Search what you want to do. There are many resources online (e.g. Statalist)

# Section 2

## Editing data in Stata

# Delete variables

- You can delete variables using the commands *drop* or *keep*
- Deleting variables is useful to
  - Simplify a very complex data-set for you to work with
  - Reduce computational time when dealing with large data-sets
  - Create temporary subsets of data for analytical purposes, like creating a table or graph
- WARNING: be careful not save the new data on top of the original

# Task 3

- Open the **cs_s0_s5_household.dta** data set

```
. use "$data\cs_s0_s5_household.dta", clear
```

- Keep the variables we will use in this excerise by typing

```
. keep hhid province district ur2012 s5cq2 s5cq4 s5cq8 s5cq15 s5c
> q23 s5bq2 s5cq22 s5cq13 s5cq17
```

- Now lets say we kept a few variables that we didnt actually needed.To drop them, type

```
. drop province s5bq2 s5cq17 s5cq15
```

# Renaming variables

- You can use the command *rename* to change the names of your variables
- Renaming is useful as
  - Can make your life easier when programming. Especially when the original variable names dont make much sense
  - It helps you remember what the variable means when a meaningful name is chosen
  - Picking a short variable name reduces time when typing it

# Task 4

- Rename all the remaining variables. Type the code bellow, one line at a time

```
. rename ur2012   urban_2012
. rename s5cq2    m_main_ws
. rename s5cq4    m_used_ws
. rename s5cq8    m_drink_ws
. rename s5cq13   earnings_sell_w
. rename s5cq22   d_affected_dis
. rename s5cq23   dis_type
```

## Generating variables

- You can use the command *generate* to create new variables
- Generating variables can be useful to
  - Change the values of a variable to a different measurement unit
  - Create a dummy variable identifying if how many observations have a given characteristic

# Task 5

- Let us create a variable that converts the number of meters to the main water source to centimeters. Type:

```
. generate  cm_main_ws = m_main_ws*100
(1,098 missing values generated)
```

- Now get descriptives for the new variable using *summarize*

# Task 5

- Let us create a dummy variable (that assumes values 0 or 1) to see if the main water source is the same as the used water source.
- First create a variable that equals zero

```
. gen  d_closest_ws        = 0
```

- Now lets replace that with 1 when it satisfies the condition that the two variables are equal. Type:

```
. replace d_closest_ws = 1 if m_main_ws == m_used_ws
(10,250 real changes made)
```

- Finally, tabulate the data using the function *tabulate*

## Labeling variables and values

- Labeling variables helps understand the variable
- Value labels indicate what each category of a categorical variable stands for
- Labeling variables and values is essential for easier understanding in the future by you and others

# Task 6

- Let us create a label for the two variables we created in Task 5

```
. label variable cm_main_ws "Cm to main water source"

. label variable d_closest_ws "Closest water source is used water
>  source"
```

- Check the variable window to see the label!

# Task 6

- We can also create labels for values with the functions *label define* and *label values*. Type:

```
. label define yes_no_lb 1 "Yes" 0 "No"
. label values d_closest_ws yes_no_lb
```

- You can see the labels if you tabulate the labeled variable or browse the data
- This is very useful for binary or categorical variables when visualizing the data

## How to share your work with your team

# Sharing work with a team

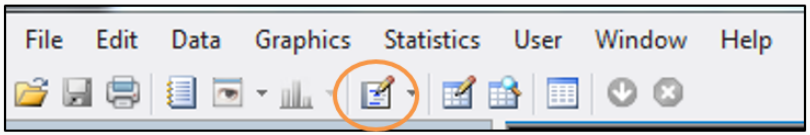
**You are asked to share your work**

- How would you share the work you have done so far?
- Send only the data set? That would be like Excel and only shares the latest version of the data
- Nowadays there's a greater demand than to share more than the latest version of the data. We need to show what we did
- This is where .do files come into the picture

## Do-files

- Open up a new do-file. Window → Do-file Editor → New Do-file Editor
- Alternatively click the shortcut highlighted below:

  - Open up a new do-file. Window -> Do-file Editor -> New Do-file Editor. Or click the shortcut highlighted below:




  - Treat the do-file similarly to how you treat the command window. But instead of copying and running one line of code at the time, a do-file lets you do that with any number of lines of code
  - Running the code in your do-file using menus: Tools -> Execute (Do). Or Ctrl+R (Windows) or this short cut:

# Task 7

- Open a new do-file. Save it!
- Type the following in your do file

```
. clear all
```

- Next, use the review window to copy to your do-file all the actions you already did:
  - Now run the do-file
  - Load the dataset
  - Keep only the variables that you need
  - Drop the ones you forgot
  - Rename all the variables
  - Create the cm to the main water source variable
  - Create the dummy if main water source is the same as the used water source
  - Label the variables and values

# Task 7

- Now lets edit the do-file!
- We just realized that the number of centimeters to the main water source doesnt make much sense. Lets edit it to the number of kilometers. Replace the code:

```
. quietly use "$data\cs_s0_s5_household.dta", clear
. quietly rename s5cq2 m_main_ws
. gen  cm_main_ws = m_main_ws*1000
(1,098 missing values generated)
```

to

```
. quietly use "$data\cs_s0_s5_household.dta", clear
. quietly rename s5cq2 m_main_ws
. gen  km_main_ws = m_main_ws/1000
(1,098 missing values generated)
```

# Comments

- Comments is the green text you have seen in the code examples
- Comments is text that Stata will ignore when running your code
- Comments is what makes the difference between instructions that are easy to follow or impossible to understand
- You can also use comments to omit certain parts of your do-file that you dont want to run anymore, but dont want to erase
    - Maybe you might need it in the future! Just be careful, keeping lots of old code in your do-file might make it messy and hard to understand.

# Different types of comments

1. */* comment *\/*
    - Used for long comments or to explain many lines of code in the following section
2. ** comment*
    - Used to explain what happens on the following few rows
3. *// comment*
    - Used to explain the same line of code

# Task 8

- Now that you know about comments, add them to your do-file!
- First, add a title and a brief explanation of what your do-file does (e.g. Stata training do-file  Uses EICV4 data to practice Stata, limiting the variables to water usage)
- Now, add a heading to every main section of your do-file (e.g. load data, keep the variables III use, create new variables, etc.)
- Finally, we realized that we actually dont need the *km_main_ws* variable now, but dont want to erase the code because we might want to use it in the future. Comment out that variable's creation.
- Run everything!

# Task 8

- Did it work?
- If you comment out the variable creation and not the labeling, you probably got an error like this:

```
.
. **** Label variables
. label variable km_main_ws "Km to main water source"
variable km_main_ws not found
r(111);
```

- To avoid this, comment out the labeling of the *km_main_ws* variable as well.

# Section 3