Stata Workshop

Roshni Khincha and Sakina Shibuya
DIME, World Bank

August, 2018

Section 1:
Basics of 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

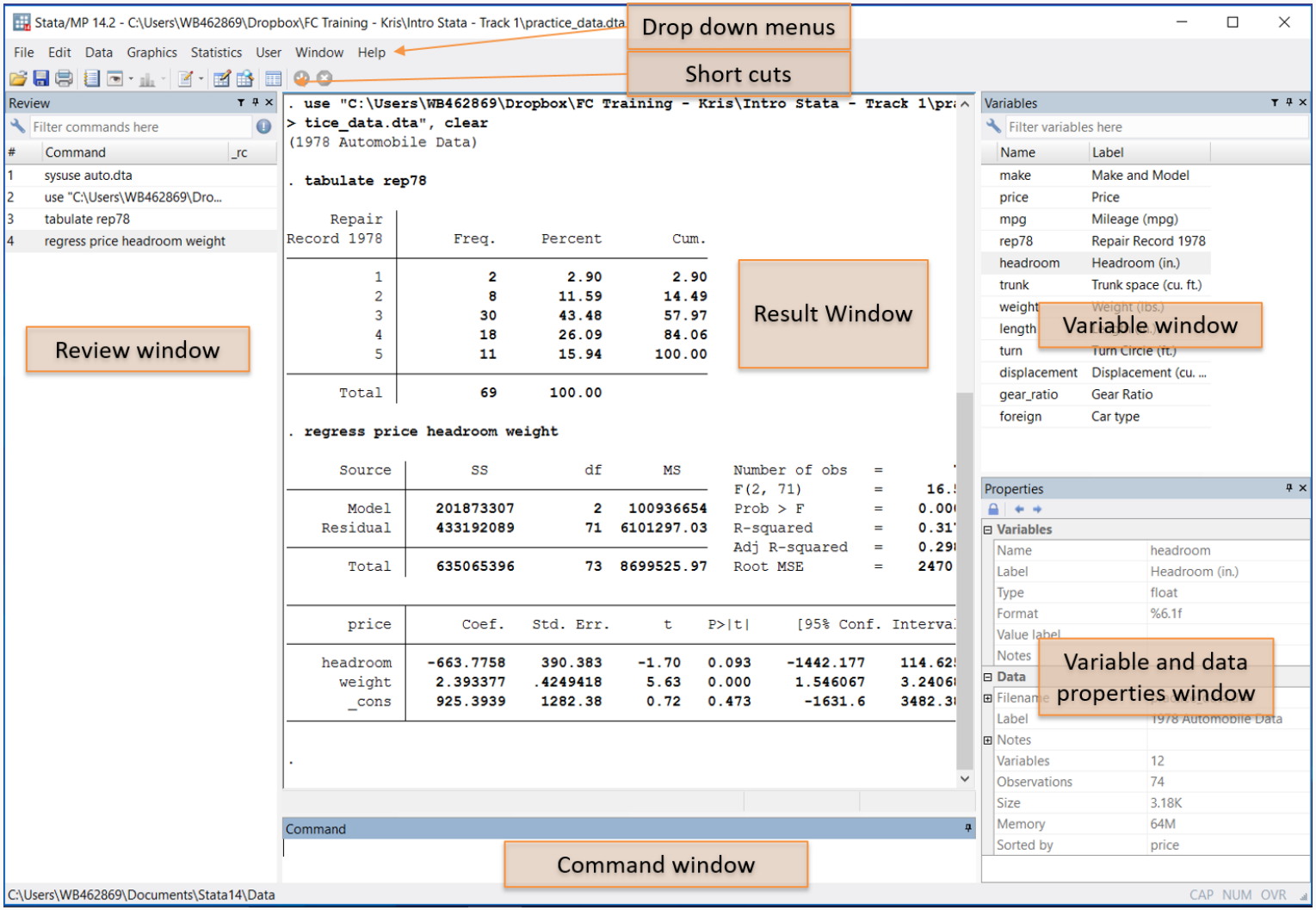
- Powerful tool with many capabilities:
  - Descriptive statistics
  - Inference statistics
  - Complex data analysis

The main reasons to use Stata

- Powerful tool with many capabilities:
  - Descriptive statistics
  - Inference statistics
  - Complex data analysis
- But it's also good for beginner programmers:
  - User friendly interface
  - Relatively easy programming language that can be learned while you're using the software

Stata interface

The Stata interface

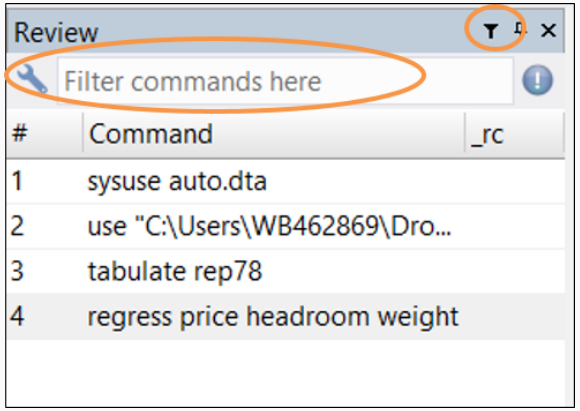


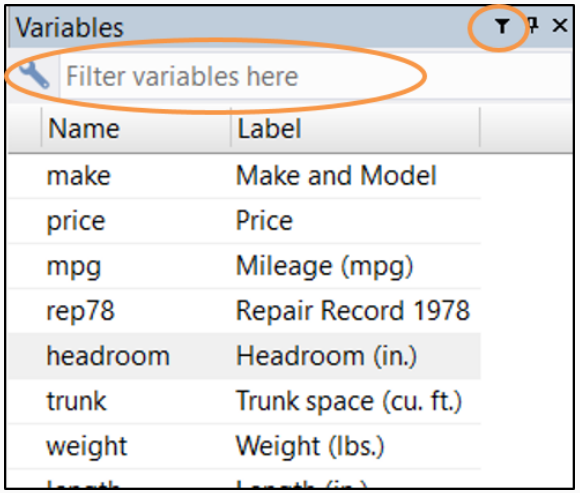
The Stata interface - Review window

- 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

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

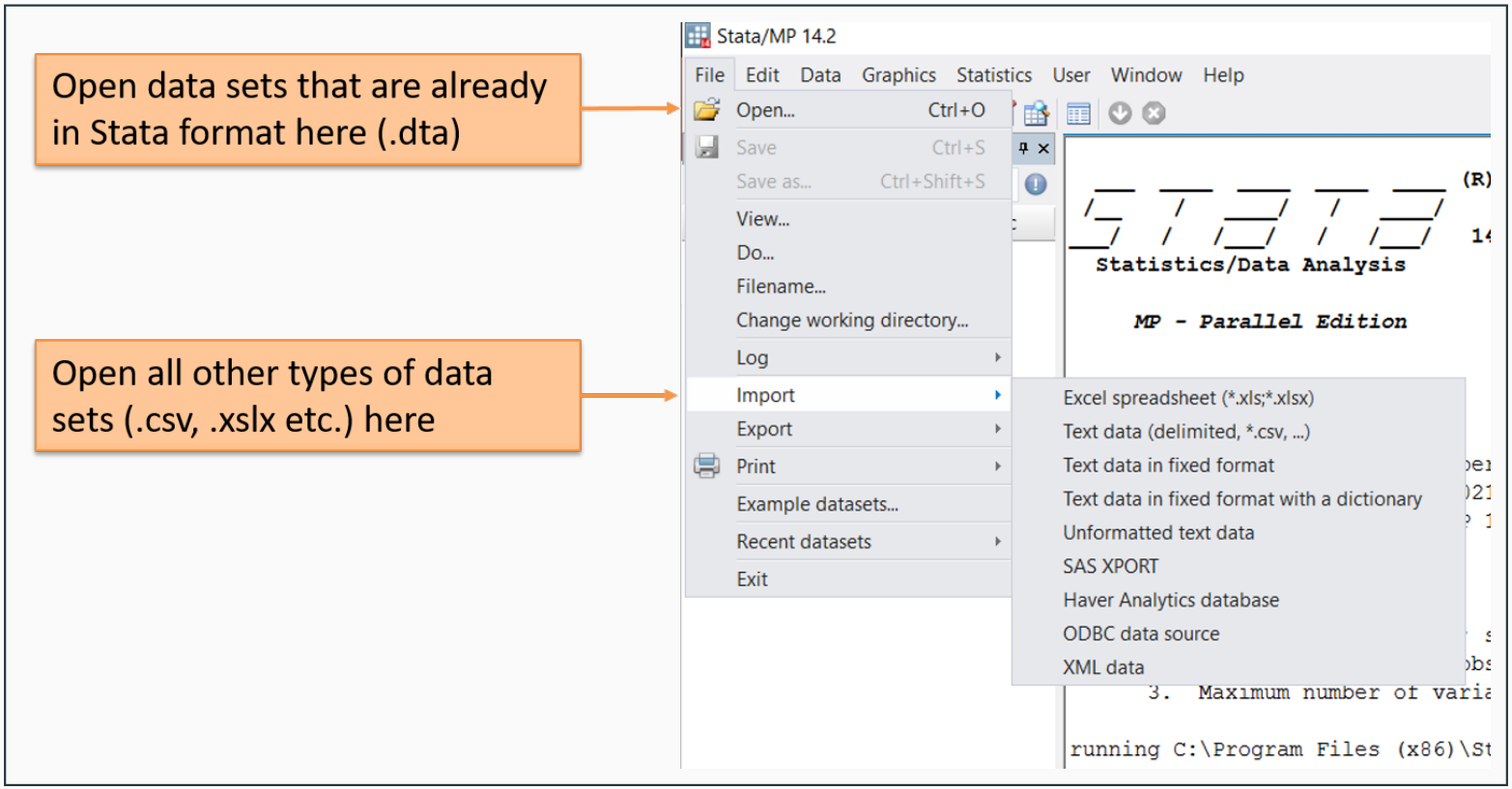
Three ways to tell stata what to do

- 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

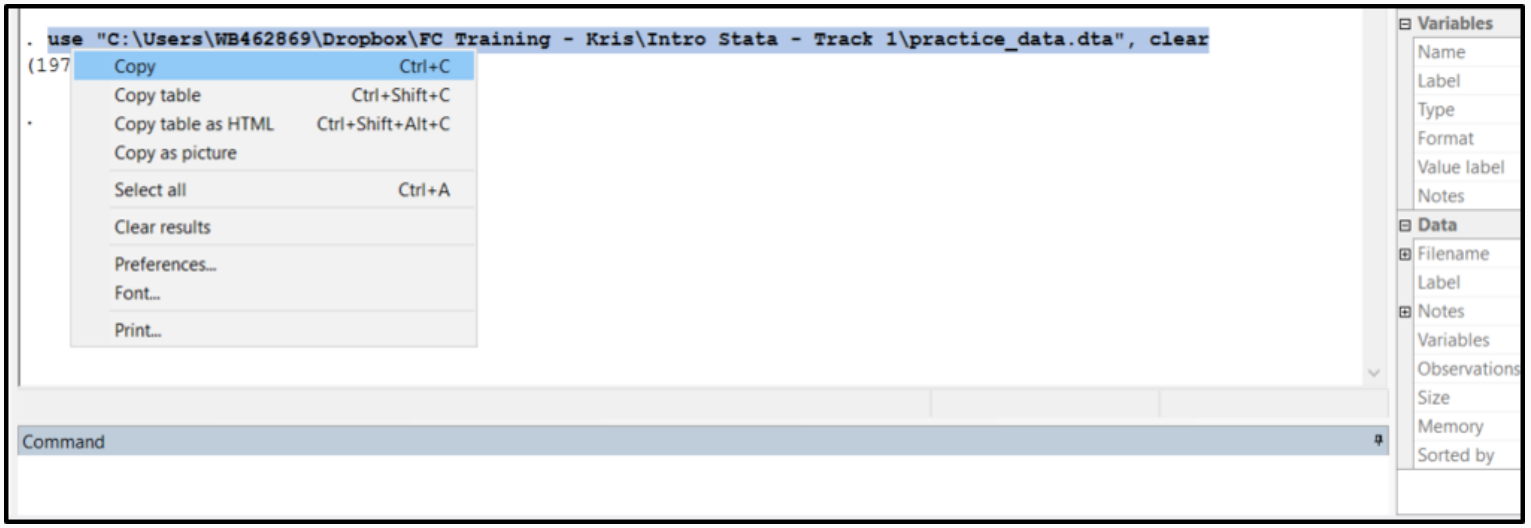
Open a dataset - menus

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 - opening datasets

1. Open Stata and then open the EICV household data set **cs_s0_s5_household.dta** using the menu: File $\rightarrow$ Open. Navigate to where you saved the material for this lab. Select the data set and click *Open*

Task 1 - opening datasets

1. Open Stata and then open the EICV household data set **cs_s0_s5_household.dta** using the menu: File $\rightarrow$ 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 $\rightarrow$ Data Editor $\rightarrow$ Data Editor Browse

Task 1 - opening datasets

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 - opening datasets

- 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 - opening datasets

- 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

Exploring a new dataset

- 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

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

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)

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

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*

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 - exploring a data set

1. Open the **cs_s0_s5_household.dta** again. Use the command prompt this time.

Task 2 - exploring a data set

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?

Task 2 - exploring a data set

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?

```
tabulate s5bq3a
summarize s5bq3a
codebook s5bq3a
```

Task 2 - exploring a data set

- Learn more about the variable *ur2012*, to learn about the proportion of urban and rural households in Rwanda

```
tabulate ur2012
```

  - Now Create a pie chart: Graphics $\rightarrow$ Pie chart, select *ur2012* as Category variable and press OK

Task 2 - exploring a data set

- Learn more about the variable *ur2012*, to learn about the proportion of urban and rural households in Rwanda

  ```
  tabulate ur2012
  ```

  - Now Create a pie chart: Graphics $\rightarrow$ 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

Editing data in Stata

Deleting 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 - deleting variables

- Open the **cs_s0_s5_household.dta** data set (use the command prompt)

Task 3 - deleting variables

- Open the **cs_s0_s5_household.dta** data set (use the command prompt)
- Keep the variables we will use in this excerise by typing

```
keep hhid province district ur2012 s5cq2 s5cq4 s5cq8 ///
s5cq15 s5cq23 s5bq2 s5cq22 s5cq13 s5cq17
```

Task 3 - deleting variables

- Open the **cs_s0_s5_household.dta** data set (use the command prompt)
- Keep the variables we will use in this excerise by typing

```
keep hhid province district ur2012 s5cq2 s5cq4 s5cq8 ///
s5cq15 s5cq23 s5bq2 s5cq22 s5cq13 s5cq17
```

- Now let's say we kept a few variables that we didn't 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 don't 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 - renaming variables

- Rename all the remaining variables. Type the code below, 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 - generating variables

- 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
```

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

Task 5 - generating variables

- 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.

Task 5 - generating variables

- 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

```
generate  d_closest_ws   = 0
```

Task 5 - generating variables

- 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

```
generate  d_closest_ws   = 0
```

- Now let's 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
```

Task 5 - generating variables

- 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

```
generate  d_closest_ws   = 0
```

- Now let's 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
```

- 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 - labeling

- 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 - labeling

- 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
```

Task 6 - labeling

- 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

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

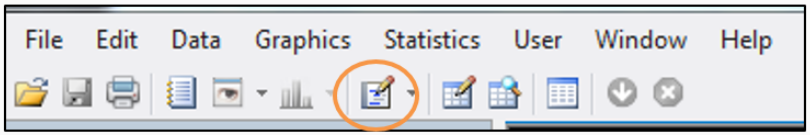
What's the fuss about do-files?

- It's 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
- A do-file works similarly to the command window. But instead of running one line of code at the time, a do-file lets you do that run any number of lines of code

Do-files

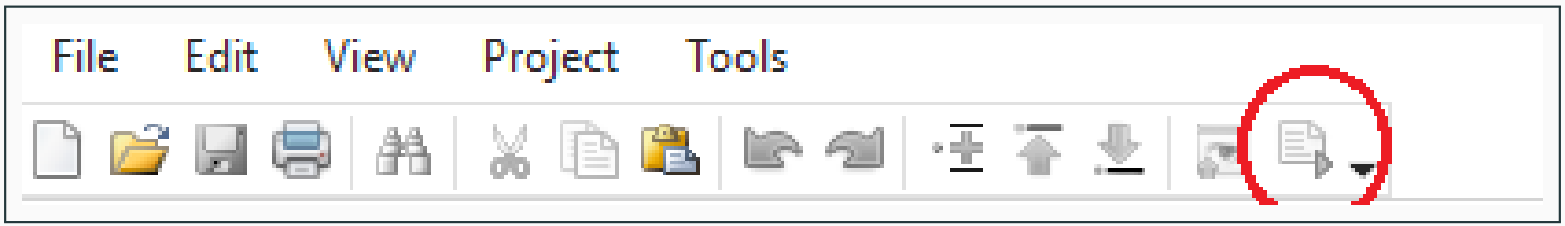
- Open up a new do-file. Window $\rightarrow$ Do-file Editor $\rightarrow$ New Do-file Editor
- Alternatively click the shortcut highlighted below:

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



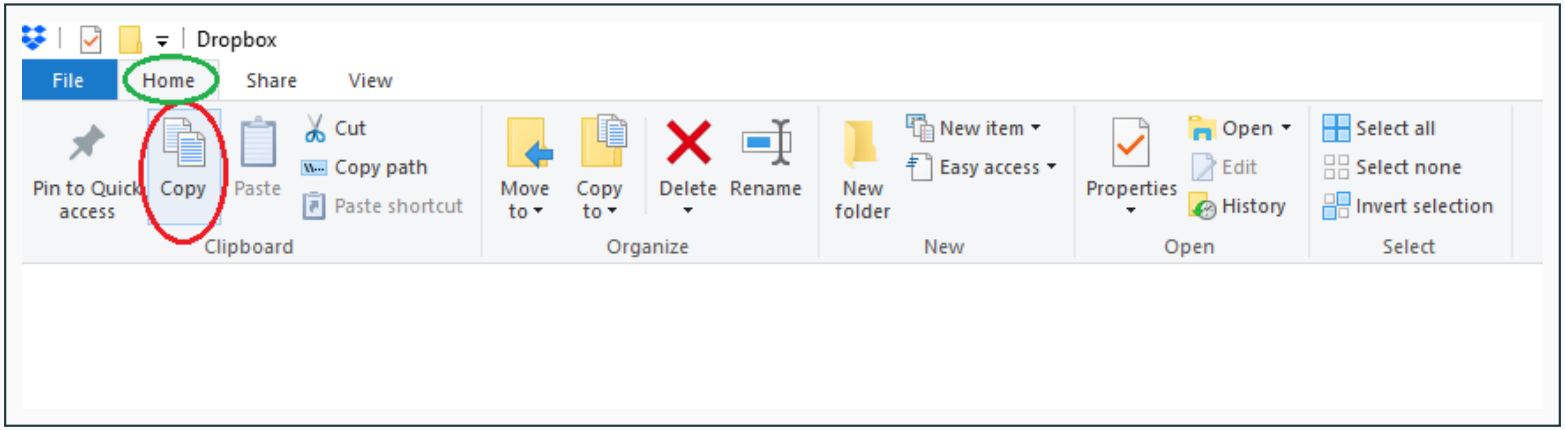
- Treat the do-file similarly to how you treat the command window. But instead of copying and running one line of code at the time, a do-file lets you do that with any number of lines of code
- Running the code in your do-file using menus: Tools -> Execute (Do). Or Ctrl+R (Windows) or this short cut:

- Running the code in your do-file using menus: Tools $\rightarrow$ Execute (Do)
- Alternatively, select Ctrl+D (Windows) on your keyboard
- Alternatively, click the shortcut highlighted below



How do you find a file path?

- File path are a integral part of sharing data which we will explain in the next slide



- When you open any folder, clicking on the **Home** button on the top right
- Then clicking on **Copy** copies the path to the folder you opened

Macros

- You need to be at least familiar with this topic as this technique is critical especially when projects grow in size
- Macros (globals, locals, scalar) save some information (text or number) that you can reference later
    - Example: We want to access files in the folder multiple times. We can store the folder location in a global and use it multiple times
- To call a global saved we use the $ symbol

Task 7 - using a do-file

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

```
clear all
```

- Open the folder where you saved the data we shared
- Copy the file path of this folder
- Create a **global** called *data* using the file path you copied

```
global  data "$dropbox\minagri_stata_training_aug2018\data"
```

Task 7 - using a do-file

- 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 using the global you created
  - 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 - using a do-file

This is how your do-file should look now

```
clear all
use "$data\cs_s0_s5_household.dta", clear
keep hhid province district ur2012 s5cq2 s5cq4 s5cq8 ///
s5cq1 s5cq23 s5bq2 s5cq22 s5cq13 s5cq15 s5cq17
drop province s5bq2 s5cq17 s5cq15
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
generate  cm_main_ws = m_main_ws*100
generate  d_closest_ws = 0
replace d_closest_ws = 1 if m_main_ws == m_used_ws
label variable cm_main_ws "Cm to main water source"
label variable d_closest_ws ///
"Closest water source is used water source"
label define yes_no_lb 1 "Yes" 0 "No"
label values d_closest_ws yes_no_lb
```

Task 7 - editing a do-file

- Now let's edit the do-file!
- We just realized that the number of centimeters to the main water source doesn't make much sense. Let's edit it to the number of kilometers. Replace the code:

```
generate  cm_main_ws = m_main_ws*1000
```

to

```
gen  km_main_ws = m_main_ws/1000
```

Also, remember to edit the label before you run the do-file!

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 don't want to run anymore, but don't 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

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

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 - commenting

- 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)

Task 8 - commenting

- 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 I'll use, create new variables, etc.)

Task 8 - commenting

- 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 I'll use, create new variables, etc.)
- Finally, we realized that we actually don't need the *km_main_ws* variable now, but don't 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 - commenting

- 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.

Saving Stata datasets

- The command for saving a Stata dataset is ***save***.
- *save* saves your data in memory in a file format called dta. This is a file that can only be read with Stata.
- The command for saving a dataset in excel and csv is ***export***.
- ***export*** is the opposite of import, and is very versatile. It lets you save data in excel, csv, sas and others. Please refer to the help file on ***export***.

Saving Stata datasets

The help file for the ***save*** command.

The ***save*** command lets you save your dataset in the dta format, which is a Stata format which can only be open with Stata.

Save data in memory to file

`save` [*filename*] [, *save_options*]

Save data in memory to file in Stata 12 format

`saveold` *filename* [, *saveold_options*]

save_options	Description
`nolabel`	omit value labels from the saved dataset
`replace`	overwrite existing dataset
`all`	save `e(sample)` with the dataset; programmer's option
`orphans`	save all value labels
`emptyok`	save dataset even if zero observations and zero variables

Saving Stata datasets

The help file for the ***export excel*** command.

The ***export excel*** command lets you save your dataset in excel.

Save data in memory to an Excel file

```
export excel [using] filename [if] [in] [, export_excel_options]
```

Save subset of variables in memory to an Excel file

```
export excel [varlist] using filename [if] [in] [, export_excel_options]
```

export_excel_options	Description
Main	
`sheet("sheetname")`	save to Excel worksheet
`cell(start)`	start (upper-left) cell in Excel to begin saving to
`sheetmodify`	modify Excel worksheet
`sheetreplace`	replace Excel worksheet
`firstrow(variables|varlabels)`	save variable names or variable labels to first row
`nolabel`	export values instead of value labels
`replace`	overwrite Excel file
Advanced	
`datestring("datetime_format")`	save dates as strings with a *datetime_format*
`missing("repval")`	save missing values as *repval*
`locale("locale")`	specify the locale used by the workbook; has no effect on

Task 9 - saving Stata datasets

Let's save the modified data as a dta file. Type...

```
save "$data\cs_s0_s5_household_modified.dta", replace
```

Task 9 - saving Stata datasets

Let's save the modified data as a dta file. Type...

```
save "$data\cs_s0_s5_household_modified.dta", replace
```

Notice that we use the ***replace*** option. This overwrites the existing file. Type the same command without ***, replace***, and see what error you get!

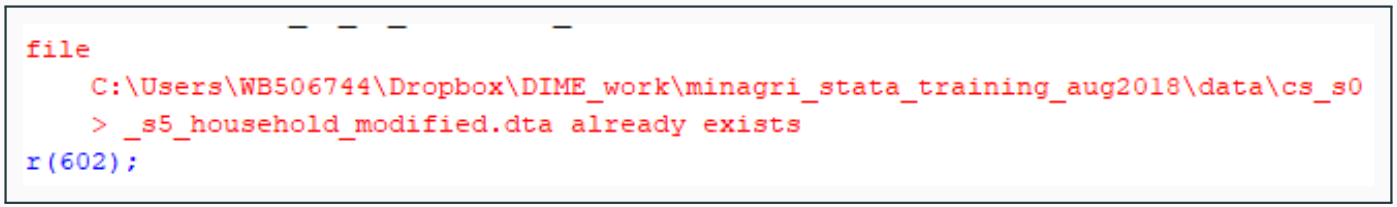
Task 9 - saving Stata datasets

Let's save the modified data as a dta file. Type...

```
save "$data\cs_s0_s5_household_modified.dta", replace
```

Notice that we use the ***replace*** option. This overwrites the existing file. Type the same command without ***, replace***, and see what error you get!

Did you get an error like this?



Task 10 - exporting Stata datasets

Now, let's save the modified data as a excel. This is helpful if you are sending the dataset to someone who does not use or have Stata. Type...

```
export excel using "$data\cs_s0_s5_household_modified.xls", replace
```

Task 10 - exporting Stata datasets

Now, let's save the modified data as a excel. This is helpful if you are sending the dataset to someone who does not use or have Stata. Type...

```
export excel using "$data\cs_s0_s5_household_modified.xls", replace
```

Open the output file. Notice that it doesn't have variable names as column names. This is very inconvenient!

Task 10 - exporting Stata datasets

Now, let's save the modified data as a excel. This is helpful if you are sending the dataset to someone who does not use or have Stata. Type...

```
export excel using "$data\cs_s0_s5_household_modified.xls", replace
```

Open the output file. Notice that it doesn't have variable names as column names. This is very inconvenient! Use an optional command, ***firstrow(variables)***.

```
export excel using "$data\cs_s0_s5_household_modified.xls", ///
replace firstrow(variables)
```

Task 10 - exporting Stata datasets

Now, let's save the modified data as a excel. This is helpful if you are sending the dataset to someone who does not use or have Stata. Type...

```
export excel using "$data\cs_s0_s5_household_modified.xls", replace
```

Open the output file. Notice that it doesn't have variable names as column names. This is very inconvenient! Use an optional command, ***firstrow(variables)***.

```
export excel using "$data\cs_s0_s5_household_modified.xls", ///
replace firstrow(variables)
```

Notice ///. This is a way to let Stata now that multiple lines constitute a single command. It's helpful when your command is getting too long on your do file. Open the newly saved excel file. You will find column names!

Section 3:
Introduction to Stata Graphics

Table gives all the details

What's happening in this regression table? What's important?

TABLE 3—MEASURES OF ACCESS TO CARE JUST BEFORE 65 AND ESTIMATED DISCONTINUITIES AT 65

	1997–2003 NHIS				1992–2003 NHIS			
	Delayed care last year		Did not get care last year		Saw doctor last year		Hospital stay last year	
	Age 63–64 (1)	RD at 65 (2)	Age 63–64 (3)	RD at 65 (4)	Age 63–64 (5)	RD at 65 (6)	Age 63–64 (7)	RD at 65 (8)
Overall sample	7.2	−1.8 (0.4)	4.9	−1.3 (0.3)	84.8	1.3 (0.7)	11.8	1.2 (0.4)
Classified by ethnicity and education:								
White non-Hispanic:								
High school dropout	11.6	−1.5 (1.1)	7.9	−0.2 (1.0)	81.7	3.1 (1.3)	14.4	1.6 (1.3)
High school graduate	7.1	0.3 (2.8)	5.5	−1.3 (2.8)	85.1	−0.4 (1.5)	12.0	0.3 (0.7)
At least some college	6.0	−1.5 (0.4)	3.7	−1.4 (0.3)	87.6	0.0 (1.3)	9.8	2.1 (0.7)
Minority:								
High school dropout	13.6	−5.3 (1.0)	11.7	−4.2 (0.9)	80.2	5.0 (2.2)	14.5	0.0 (1.4)
High school graduate	4.3	−3.8 (3.2)	1.2	1.5 (3.7)	84.8	1.9 (2.7)	11.4	1.8 (1.4)
At least some college	5.4	−0.6 (1.1)	4.8	−0.2 (0.8)	85.0	3.7 (3.9)	9.5	0.7 (2.0)
Classified by ethnicity only:								
White non-Hispanic	6.9	−1.6 (0.4)	4.4	−1.2 (0.3)	85.3	0.6 (0.8)	11.6	1.3 (0.5)
Black non-Hispanic (all)	7.3	−1.9 (1.1)	6.4	−0.3 (1.1)	84.2	3.6 (1.9)	14.4	0.5 (1.1)
Hispanic (all)	11.1	−4.9 (0.8)	9.3	−3.8 (0.7)	79.4	8.2 (0.8)	11.8	1.0 (1.6)

Note: Entries in odd numbered columns are mean of variable in column heading among people ages 63–64. Entries in even numbered columns are estimated regression discontinuities at age 65, from models that include linear control for age interacted with dummy for age 65 or older (columns 2 and 4) or quadratic control for age, interacted with dummy for age 65 and older (columns 6 and 8). Other controls in models include indicators for gender, race/ethnicity, education, region, and sample year. Sample in columns 1–4 is pooled 1997–2003 NHIS. Sample in columns 5–8 is pooled 1992–2003 NHIS. Samples for regression models include people ages 55–75 only. Standard errors (in parentheses) are clustered by quarter of age.

But figures *tell the story*

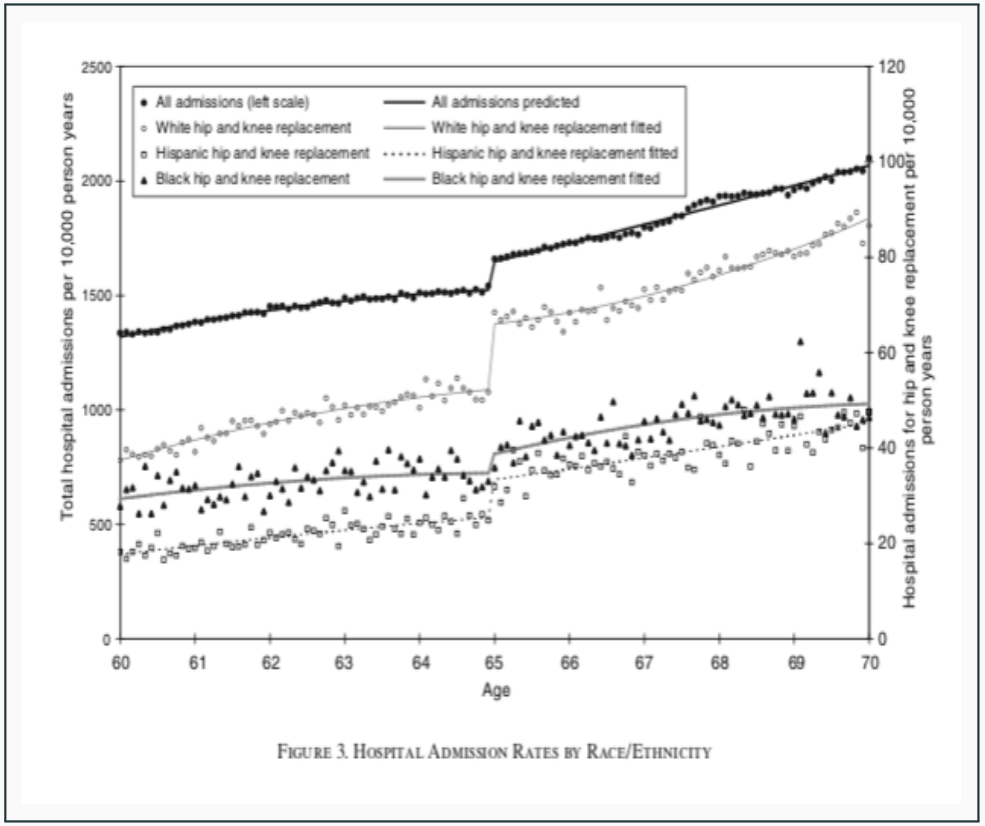
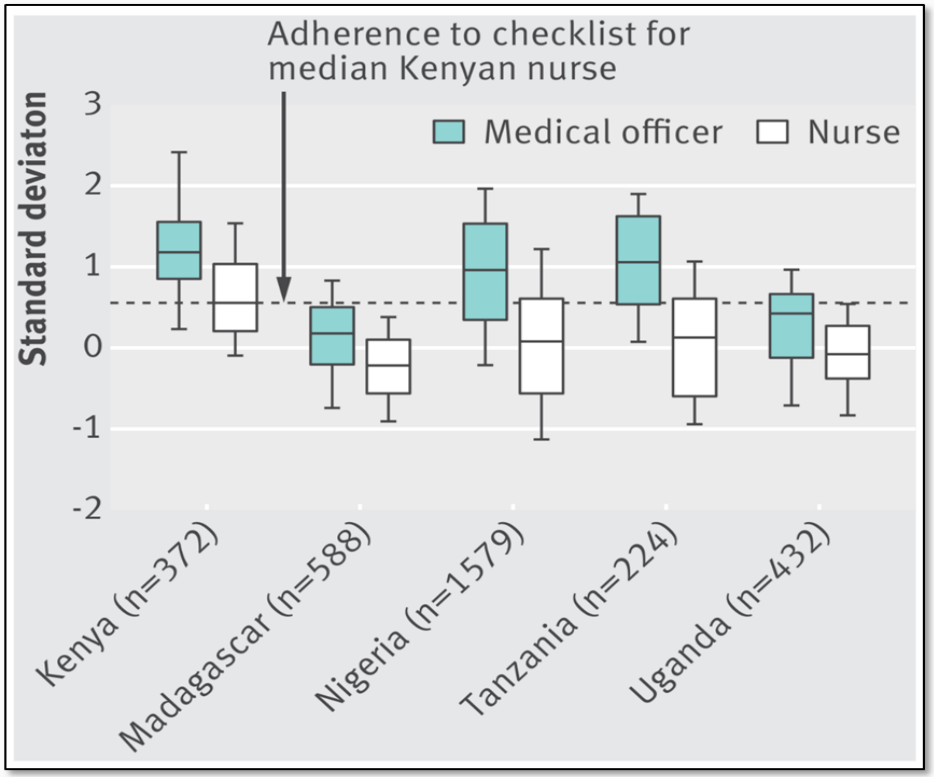


FIGURE 3. HOSPITAL ADMISSION RATES BY RACE/ETHNICITY

- This is the data that generates those estimates.
- You can see exactly what is happening very quickly!
- Even more importantly: **Your eyes are naturally drawn to the story!**

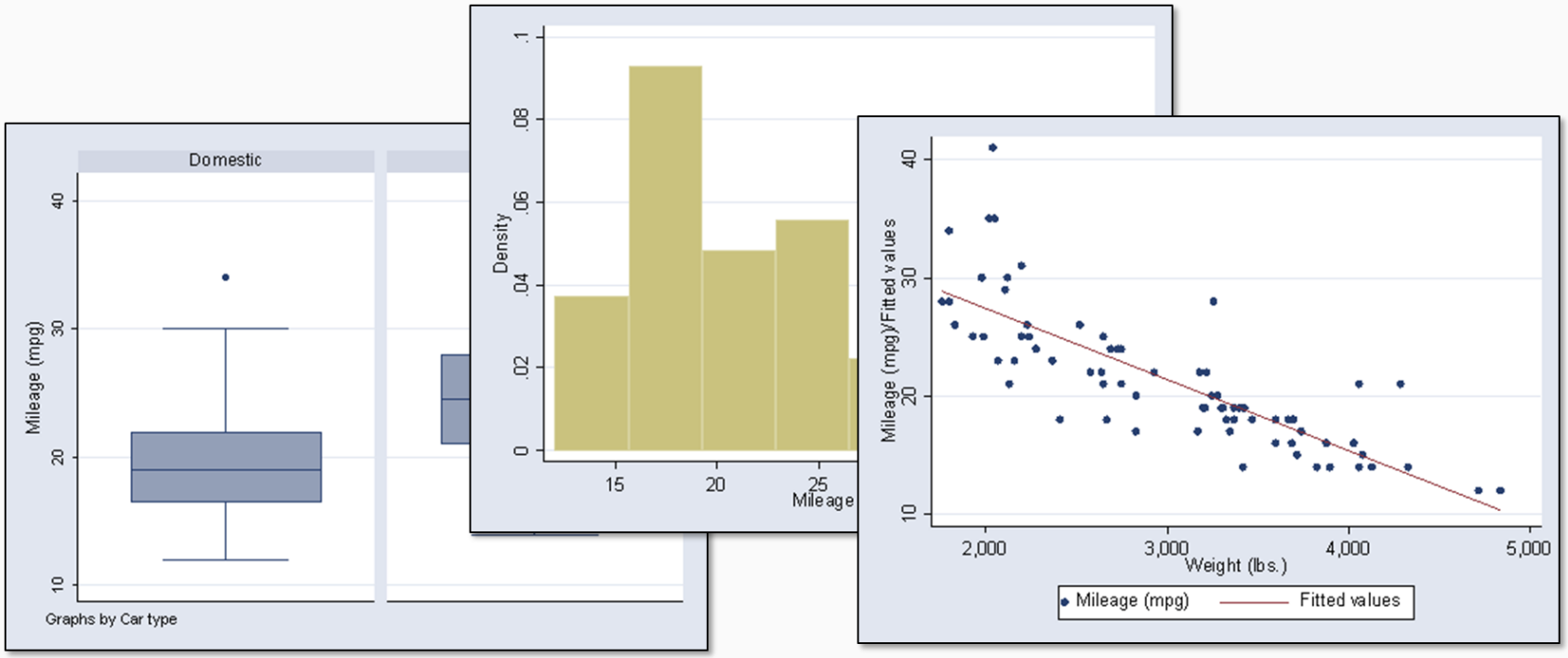
Example: comparing means



- What is the main story in this graph?
- We need more context to say something detailed about this, but what has the person creating the graph highlighted for us?

Stata default graphs

- This is what a Stata graph looks like with very minimal customizing using optional commands
- Notice that there is no graph title. They are still informative, but need much improvement
- We will not go too deep to editing a Stata graph today, but I'll show you have to make a graph and make some edits for effective data visualization



Stata has three core built-in graph functions.

[graph *graphtype*]
graphs which plot one or more variables on one axis

[twoway *graphtype*]
graphs which plot two variables together on an x and y axis

twoway_options is a set of optional commands that can be applied to all twoway graphs.
[histogram], [kdensity], [lowess]
Essential distributional commands

The other graph commands are implemented in terms of **graph**, which provides the following capabilities:

Command	Description
graph bar	bar charts
graph pie	pie charts
graph dot	dot charts
graph matrix	scatterplot matrices
graph twoway	twoway (*y*-*x*) graphs, including
graph twoway scatter	scatterplots
graph twoway line	line plots
graph twoway function	function plots
graph twoway histogram	histograms
graph twoway *	more

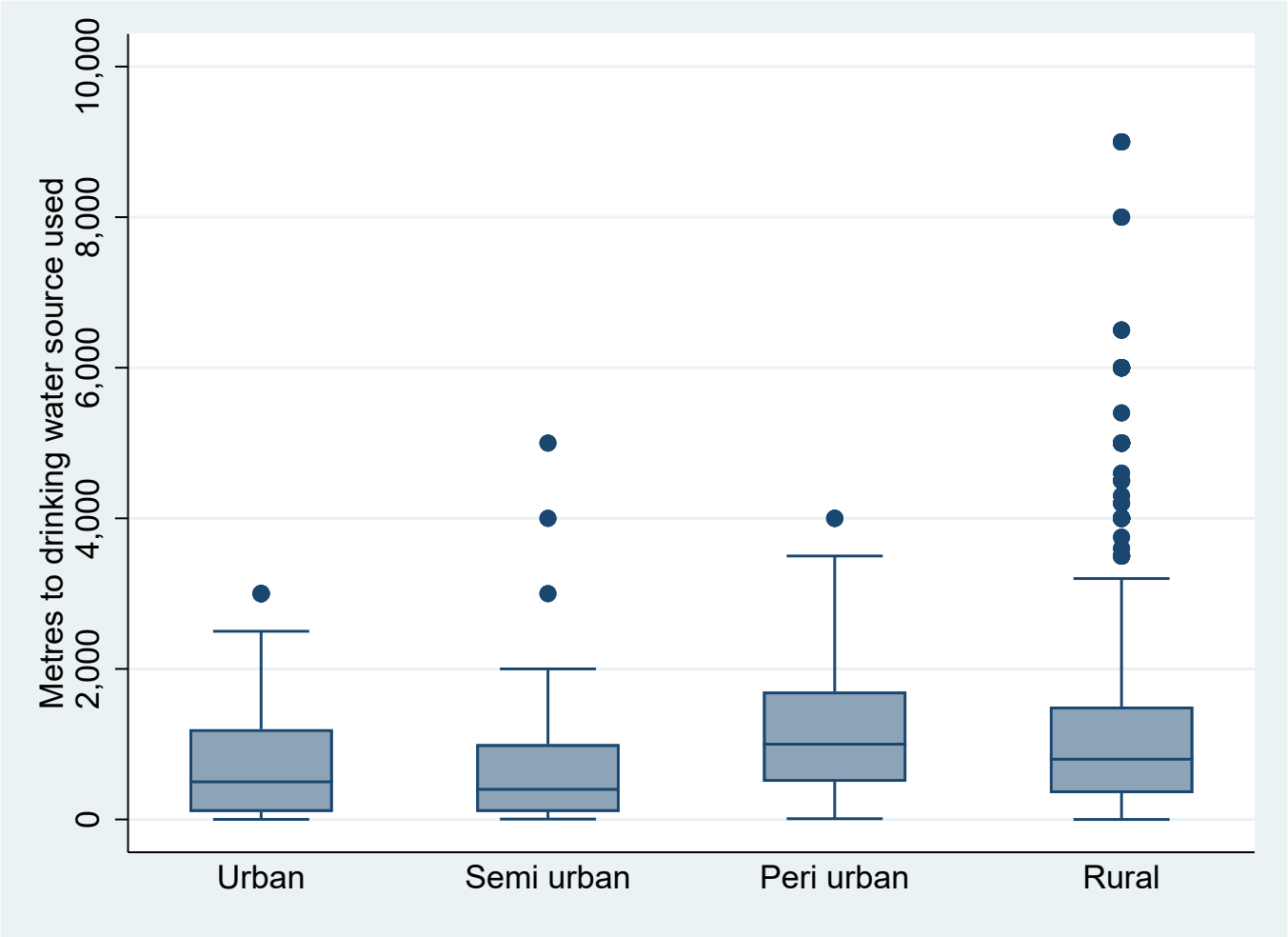
Smoothing and densities:

Command	Description
kdensity	kernel density estimation, univariate
lowess	lowess smoothing
lpoly	local polynomial smoothing

Box plot

Box plot

Let's make a a box plot like the one below using the variable, **m_drink_ws**. Notice a box plot is an example of a oneway graph.



Box plot

Let's make a box plot from your do file.

1. Open "$data_cs_s0_s5_household_modified.xls"
2. Type ***search box plots*** in the command window to find out what command to be used. ***search*** is a more general search through help files and other Stata resources.

Box plot

Let's make a box plot from your do file.

1. Open "$data_cs_s0_s5_household_modified.xls"
2. Type ***search box plots*** in the command window to find out what command to be used. ***search*** is a more general search through help files and other Stata resources.
3. The command should look like the following. Run from the do file.

```
graph box m_drink_ws
```

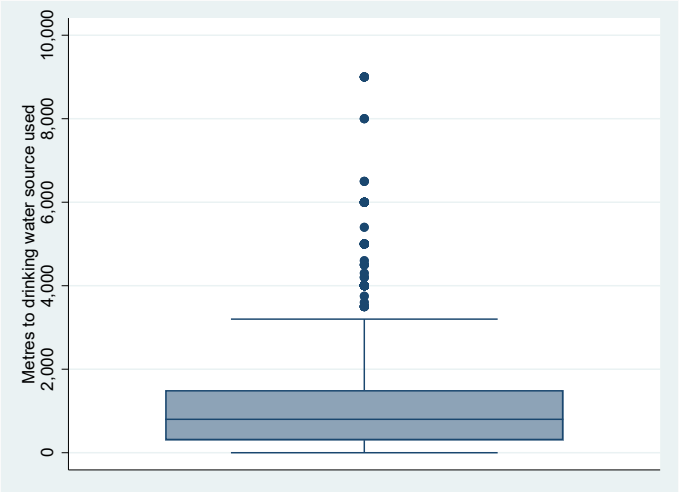
Box plot

Let's make a box plot from your do file.

1. Open "$data_cs_s0_s5_household_modified.xls"
2. Type ***search box plots*** in the command window to find out what command to be used. ***search*** is a more general search through help files and other Stata resources.
3. The command should look like the following. Run from the do file.

```
graph box m_drink_ws
```

4. Notice the difference from earlier?



Box plot

Now, let's make multiple box plots by the residential environment, **urban_2012**.

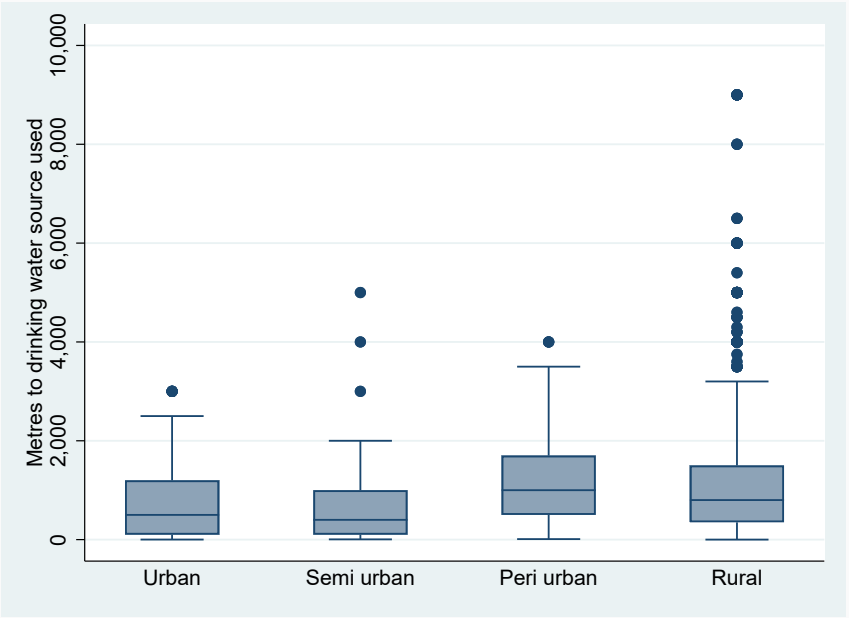
1. Type ***search box plots*** to see how to achieve this.

Box plot

Now, let's make multiple box plots by the residential environment, **urban_2012**.

1. Type ***search box plots*** to see how to achieve this.
2. The optional commant, ***over()*** can do this. Run the new ***graph box*** command with the ***over*** option.

```
graph box m_drink_ws, over(urban_2012)
```

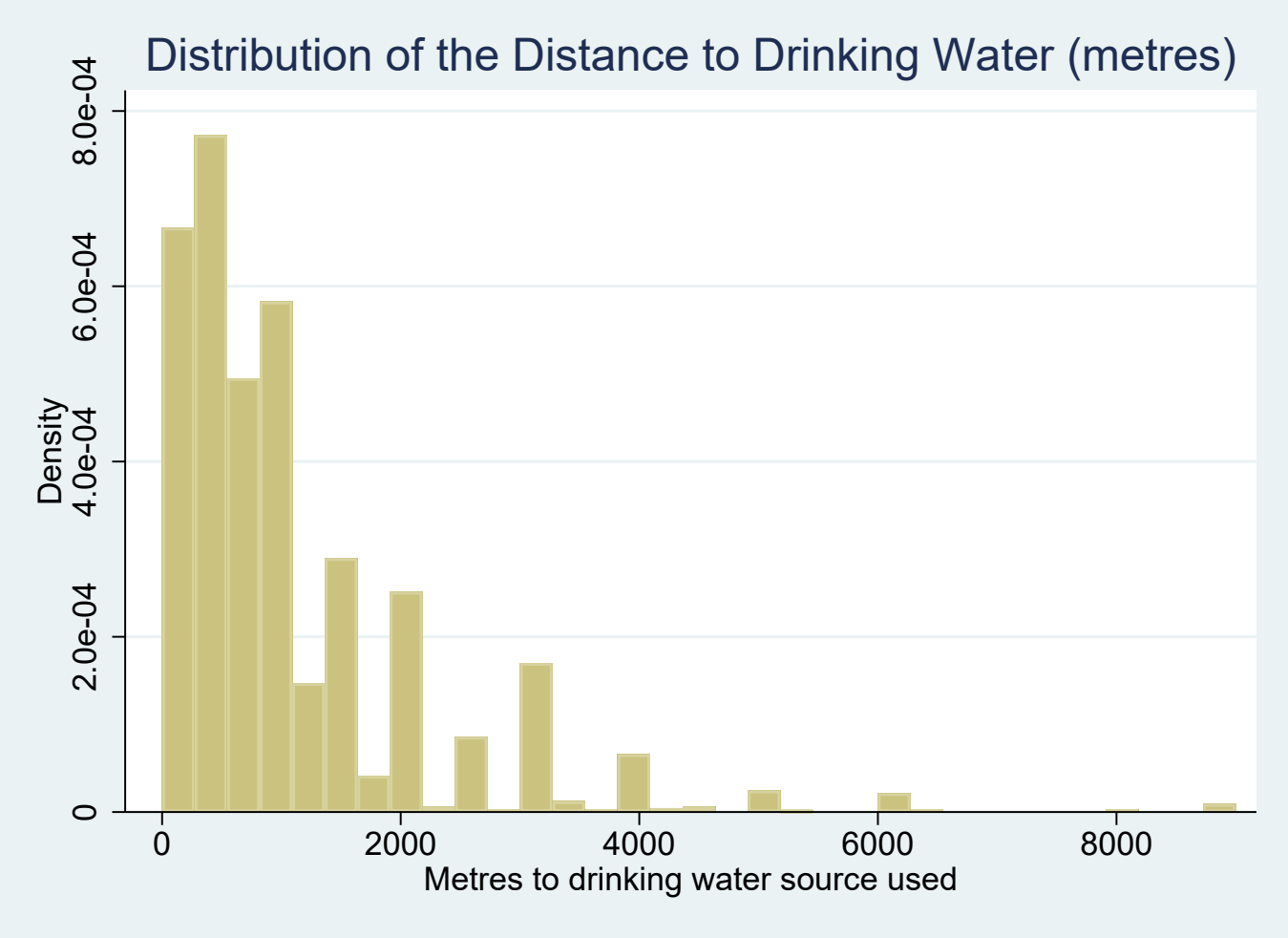


Stata graph exercise 2

Histogram

Histogram

Let's make a histogram like the one below using the variable, **m_drink_ws**. Notice that a histogram is an example of a twoway graph.



Histogram

Let's make a histogram from your do file.

1. Type ***help histogram*** in the command window to find out what command to be used.

Histogram

Let's make a histogram from your do file.

1. Type ***help histogram*** in the command window to find out what command to be used.
2. The command should look like the following. Run from the do file.

```
histogram m_drink_ws
```

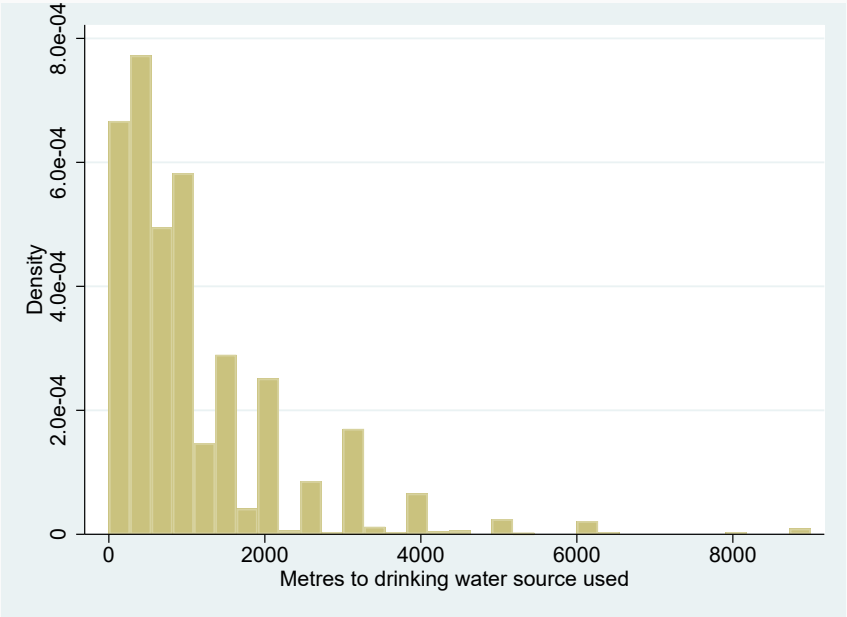
Histogram

Let's make a histogram from your do file.

1. Type ***help histogram*** in the command window to find out what command to be used.
2. The command should look like the following. Run from the do file.

```
histogram m_drink_ws
```

3. Notice the difference from earlier? We need the title.



Histogram

Now, let's add the title. You can also choose your own title that is informative. Notice in general a good title is informative but short.

Histogram

Now, let's add the title. You can also choose your own title that is informative. Notice in general a good title is informative but short.

1. The optional command, ***title()*** can do this. Run the new ***histogram*** command with the ***title*** option.

```
histogram m_drink_ws, ///
title("Distribution of the Distance to Drinking Water (metres)")
```

Histogram

Now, let's add the title. You can also choose your own title that is informative. Notice in general a good title is informative but short.

1. The optional command, ***title()*** can do this. Run the new ***histogram*** command with the ***title*** option.

```
histogram m_drink_ws, ///
title("Distribution of the Distance to Drinking Water (metres)")
```

Histogram

Now, let's add the title. You can also choose your own title that is informative. Notice in general a good title is informative but short.

1. The optional command, ***title()*** can do this. Run the new ***histogram*** command with the ***title*** option.

```
histogram m_drink_ws, ///
title("Distribution of the Distance to Drinking Water (metres)")
```

2. ***help twoway_options*** to find out more about the ***title*** option and more.

Histogram

I wanted to start today by adding onto the histrogram we made yesterday.

Histogram

I wanted to start today by adding onto the histrogram we made yesterday. The optional command, ***by()*** splits by a categorical variable. Try:

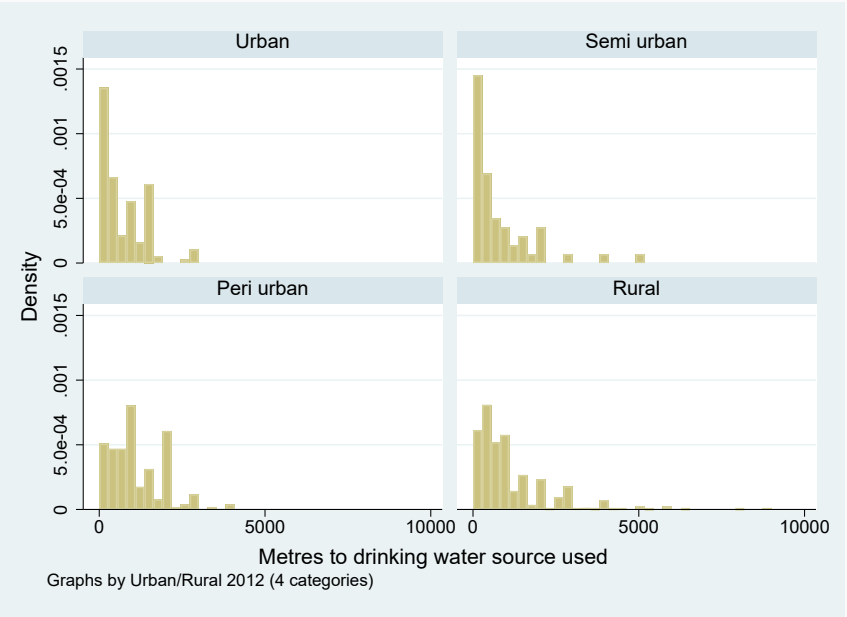
```
histogram m_drink_ws, by(urban_2012)
```

Histogram

I wanted to start today by adding onto the histrogram we made yesterday. The optional command, ***by()*** splits by a categorical variable. Try:

```
histogram m_drink_ws, by(urban_2012)
```

Does your graph look like this?

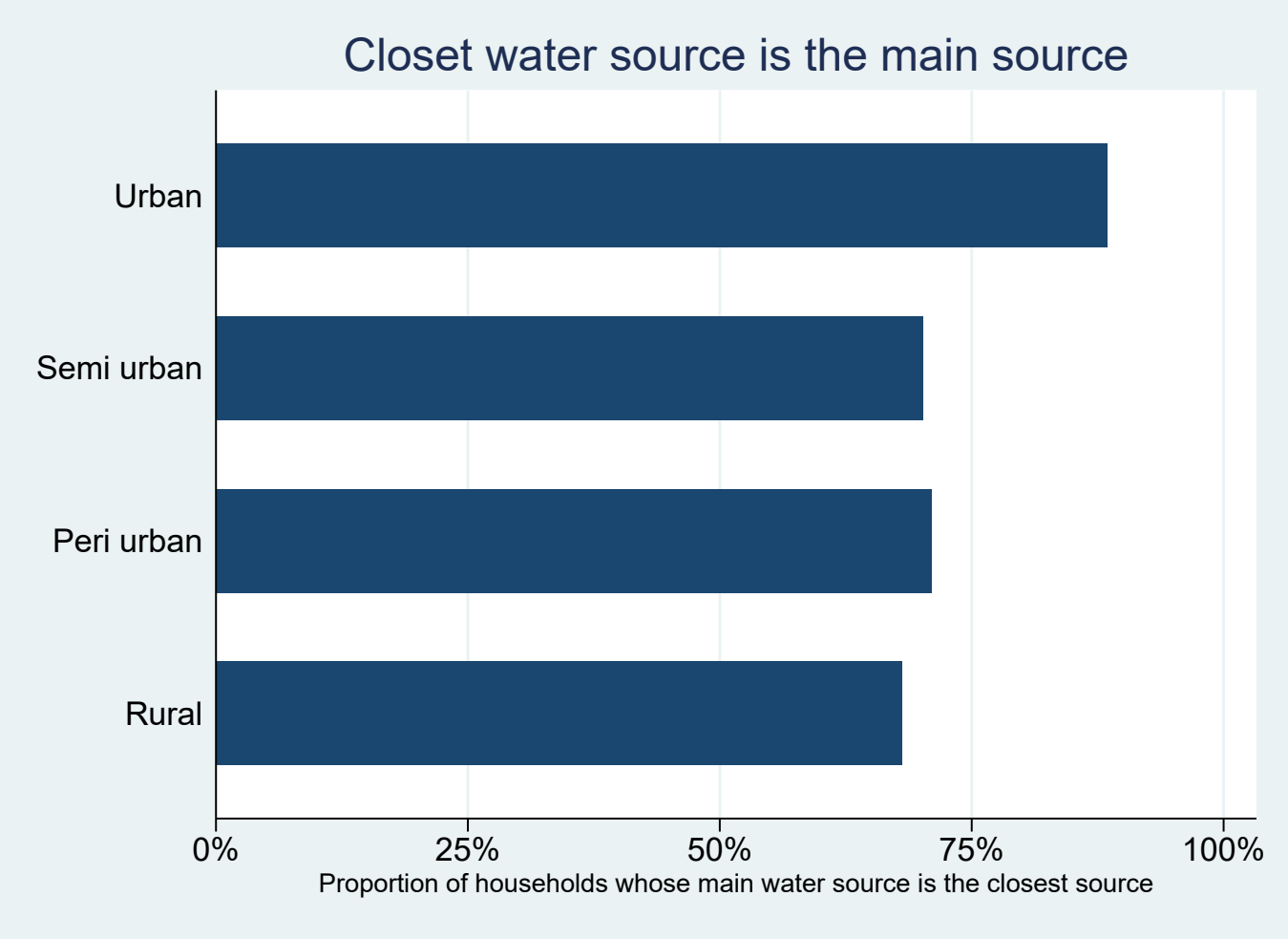


Stata graph exercise 3

Bar graph

Bar graph

Let's make a bar graph like the one below using the variable, **d_closest_ws** and **urban_2012**.



Bar graph

Let's make a bar graph from your do file.

1. Type ***help graph bar*** in the command window to find out what command to be used.

Bar graph

Let's make a bar graph from your do file.

1. Type ***help graph bar*** in the command window to find out what command to be used.
2. Note that you can make vertical or horizontal bars with this commands. The basic command should look like this.

```
graph hbar d_closest_ws
```

Bar graph

Let's make a bar graph from your do file.

1. Type ***help graph bar*** in the command window to find out what command to be used.
2. Note that you can make vertical or horizontal bars with this commands. The basic command should look like this.

```
graph hbar d_closest_ws
```

3. Recall the ***over*** option.

Bar graph

Let's make a bar graph from your do file.

1. Type ***help graph bar*** in the command window to find out what command to be used.
2. Note that you can make vertical or horizontal bars with this commands. The basic command should look like this.

```
graph hbar d_closest_ws
```

3. Recall the ***over*** option.
4. It should look something like this.

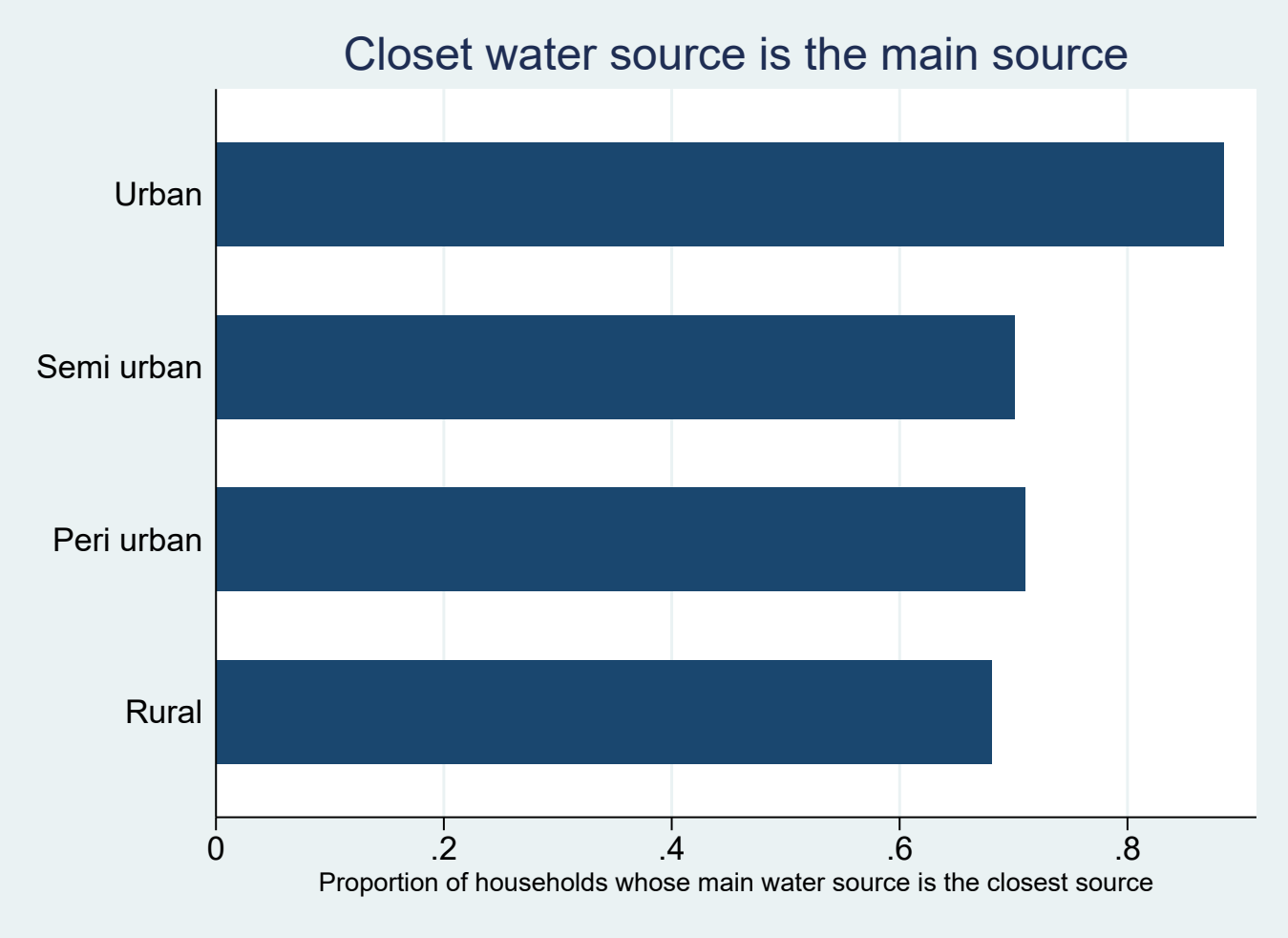
```
graph hbar d_closest_ws , over(urban_2012)
```

Bar graph

Now we need to add the main and y axis titles. Recall yesterday's lesson and try on your own.

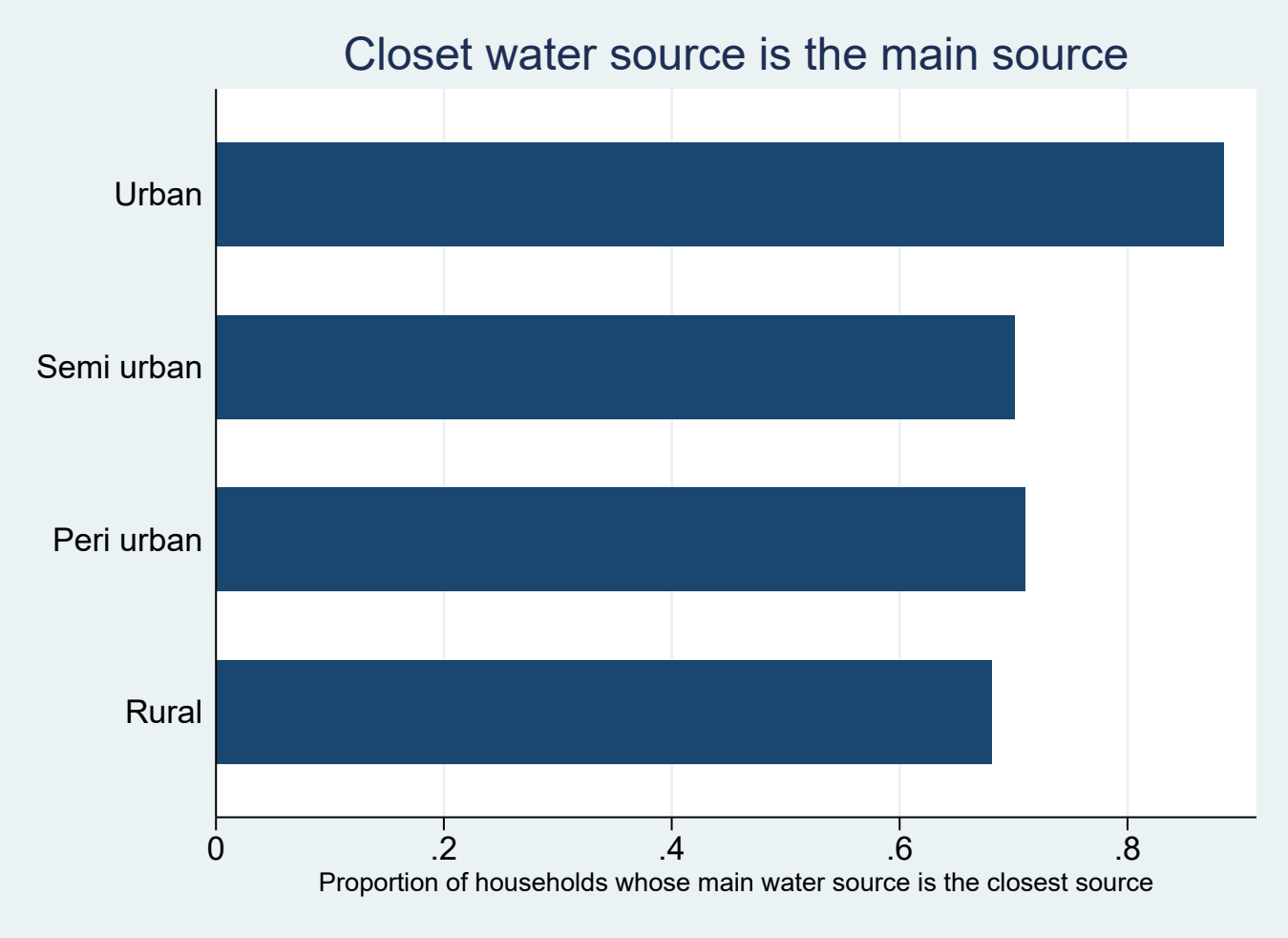
Bar graph

Now we need to add the main and y axis titles. Recall yesterday's lesson and try on your own. Does your command and graph look like this?



Bar graph

Now we need to add the main and y axis titles. Recall yesterday's lesson and try on your own. Does your command and graph look like this?



Bar graph

The y axis has unintuitive labels. This is proportion. So let's convert to percentage. We can achieve this with the optional command, ***ylabel***.

Type ***help axis_label_options*** in the command window to learn how to use this.

Bar graph

The y axis has unintuitive labels. This is proportion. So let's convert to percentage. We can achieve this with the optional command, ***ylabel***.

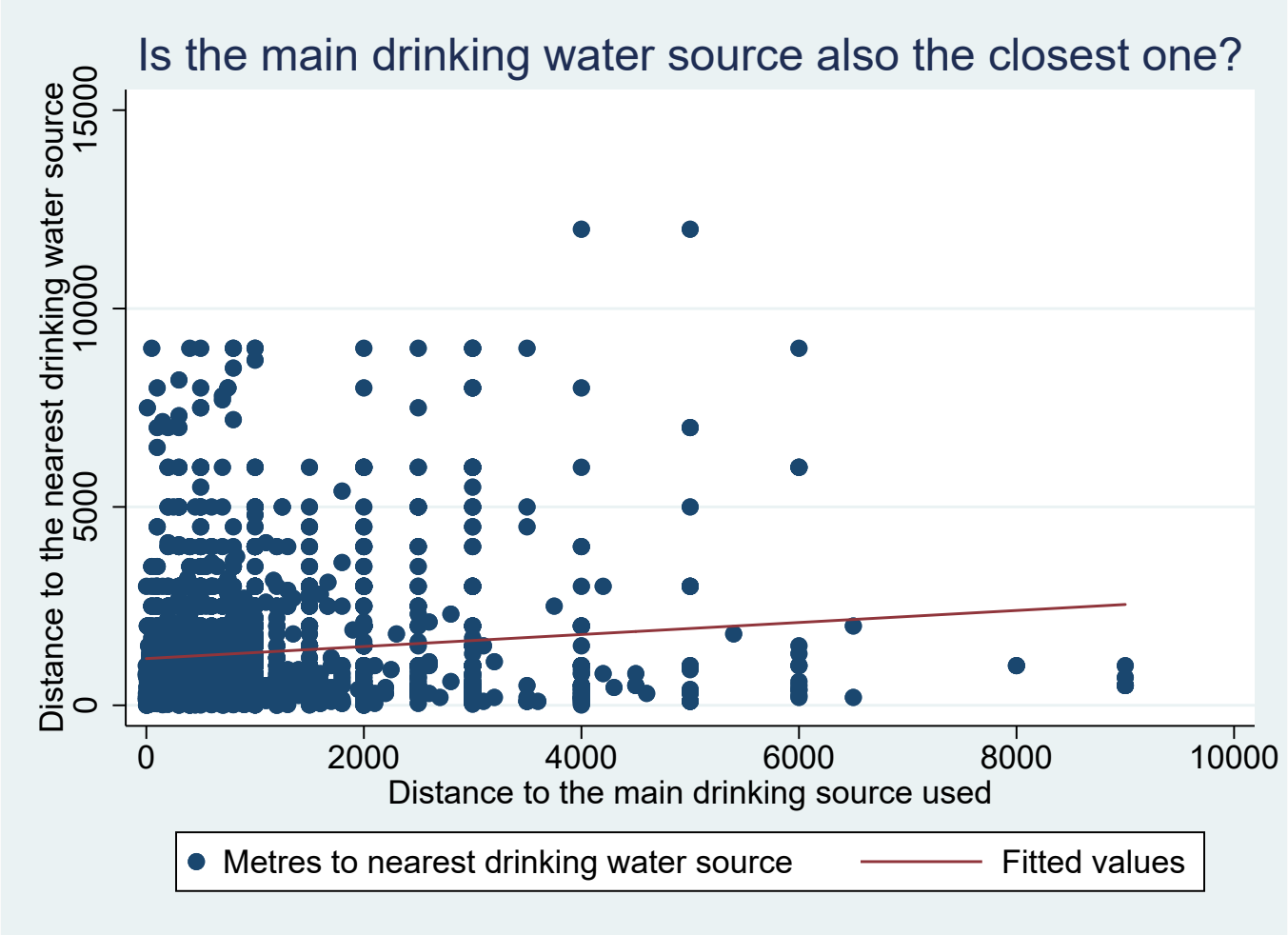
Type ***help axis_label_options*** in the command window to learn how to use this. Your command should look something like this.

```
graph hbar d_closest_ws , over(urban_2012) ///
        title("Closet water source is the main source") ///
        ylabel(0 "0%" .25 "25%" .5 "50%" .75 "75%" 1 "100%") ///
        ytitle("Proportion of households", size(small))
```

Scatter plot

Scatter plot

Let's make a scatter plot with a fitted line like the one below using the variable, **m_drink_ws** and **m_used_ws**. Notice that a scatter plot with a fitted line is an example of a twoway graph.



Scatter plot

Let's make a scatter plot from your do file.

1. Type ***help scatter*** in the command window to find out what command to be used.

Scatter plot

Let's make a scatter plot from your do file.

1. Type ***help scatter*** in the command window to find out what command to be used.
2. The command should look like the following. Run from the do file.

```
scatter m_used_ws m_drink_ws
```

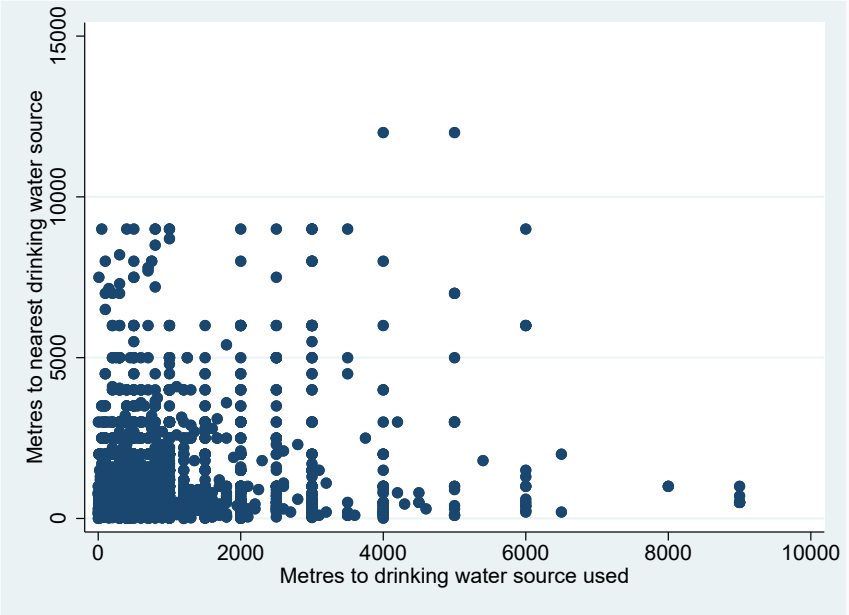
Scatter plot

Let's make a scatter plot from your do file.

1. Type ***help scatter*** in the command window to find out what command to be used.
2. The command should look like the following. Run from the do file.

```
scatter m_used_ws m_drink_ws
```

3. Notice the difference from earlier? No fitted line!



Scatter plot

Let's add a fitted line. Type ***help lfit*** to learn how to do this.

Scatter plot

Let's add a fitted line. Type ***help lfit*** to learn how to do this.

1. You may notice that this is an entirely different command. Stata can actually overlay multiple twoway graphs. To do this, run the following command. Notice that || is a way to overlay the graphs.

```
scatter m_used_ws m_drink_ws  || ///
lfit m_drink_ws m_used_ws
```

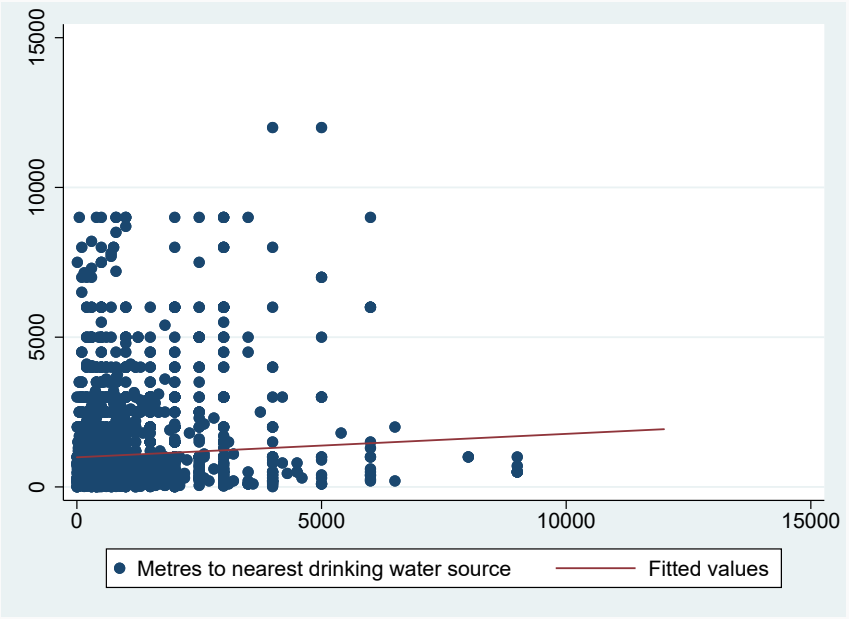
Scatter plot

Let's add a fitted line. Type ***help lfit*** to learn how to do this.

1. You may notice that this is an entirely different command. Stata can actually overlay multiple twoway graphs. To do this, run the following command. Notice that || is a way to overlay the graphs.

```
scatter m_used_ws m_drink_ws  || ///
lfit m_drink_ws m_used_ws
```

2. Notice the difference from earlier? No main title and x and y titles!



Scatter plot

This is because the fitted line is a linear prediction and no longer represents the raw distance values. But we can simply add on titles that can be helpful for the graph's intended audience.

Scatter plot

This is because the fitted line is a linear prediction and no longer represents the raw distance values. But we can simply add on titles that can be helpful for the graph's intended audience.

Recall the ***twoway_options***, and the ***title()*** option. You can use the same option and very similar options called ***xtitle()*** and ***ytitle()***.

```
scatter m_used_ws m_drink_ws  || ///
lfit m_used_ws m_drink_ws, ///
        ytitle("Distance to the nearest drinking water source") ///
        xtitle("Distance to the main drinking source used") ///
        title("Is the main drinking water also the closest source?")
```

Saving and combining graphs

Saving and combining graphs

Let's save all 5 graphs we made during this training.

Saving and combining graphs

Let's save all 5 graphs we made during this training.

To do so, add ***graph save*** after each of your graphs like the following.

```
histogram m_drink_ws, ///
title("Distribution of the Distance to Drinking Water (metres)")
graph save "$script/histogram.gph", replace
```

Notice that you need to specify where you want save it, and how you want to name it.

Saving and combining graphs

Stata can also combine multiple graphs and output as a single figure.

Saving and combining graphs

Stata can also combine multiple graphs and output as a single figure. To do so, use the command, ***graph combine***.

Check the help file to learn how to use it.

Saving and combining graphs

Stata can also combine multiple graphs and output as a single figure. To do so, use the command, ***graph combine***.

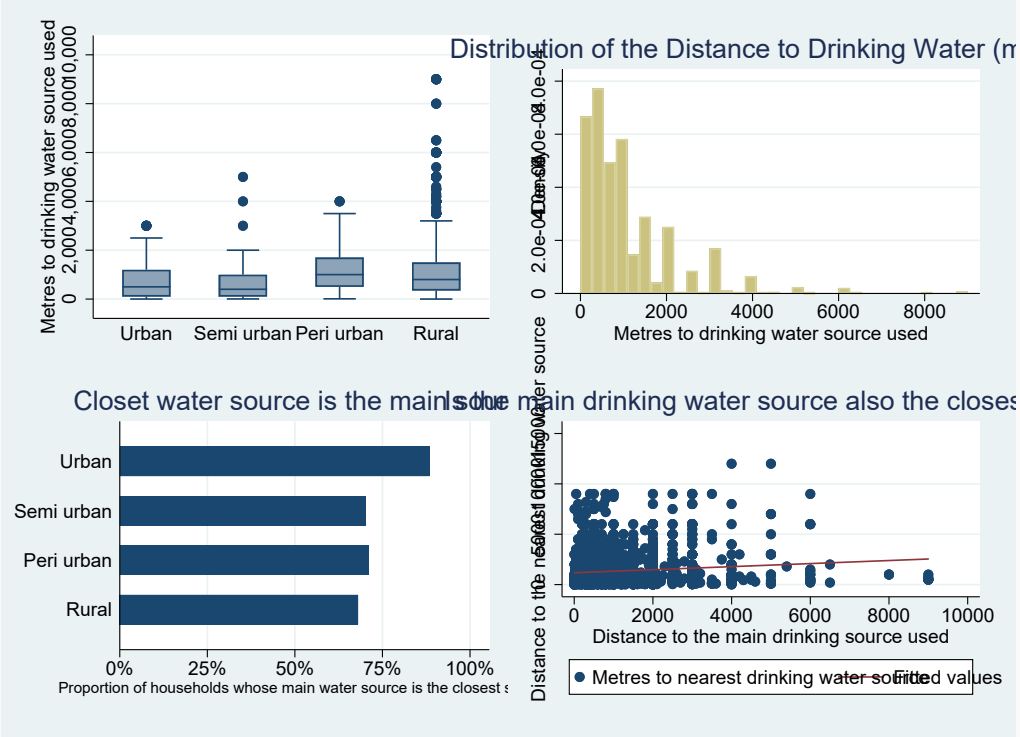
Check the help file to learn how to use it.

This is what your command should look like.

```
graph combine ///
        "$script/box.gph" ///
        "$script/histogram.gph" ///
        "$script/graph_bar.gph" ///
        "$script/scatter.gph"
```

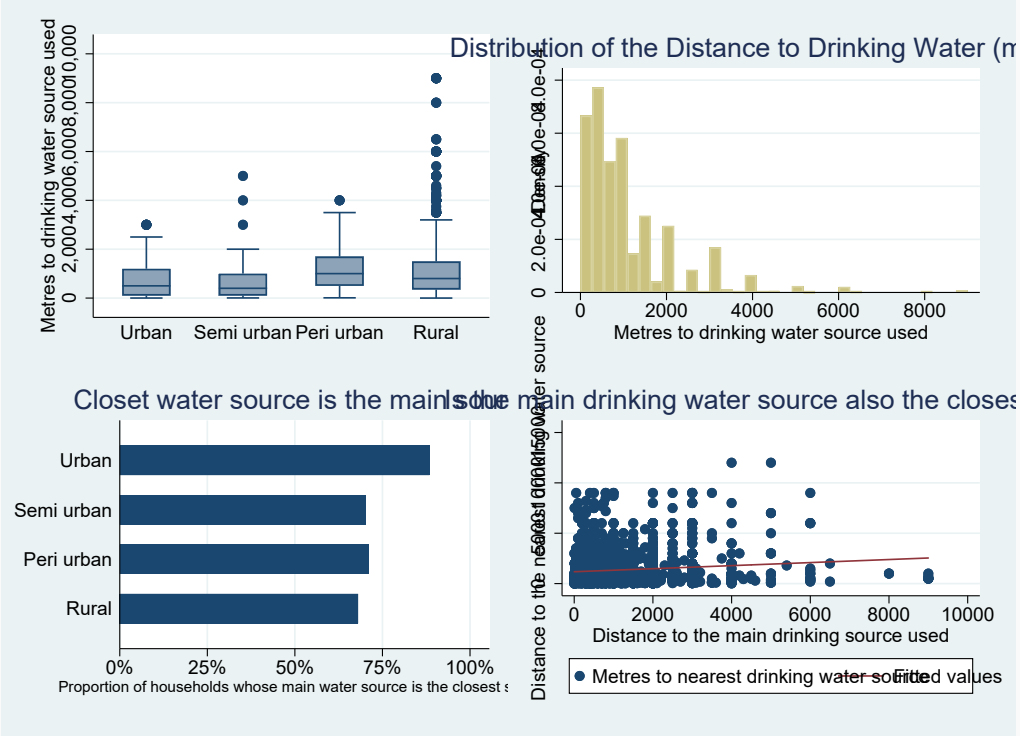
Saving and combining graphs

Does yours look like this?



Saving and combining graphs

Does yours look like this?

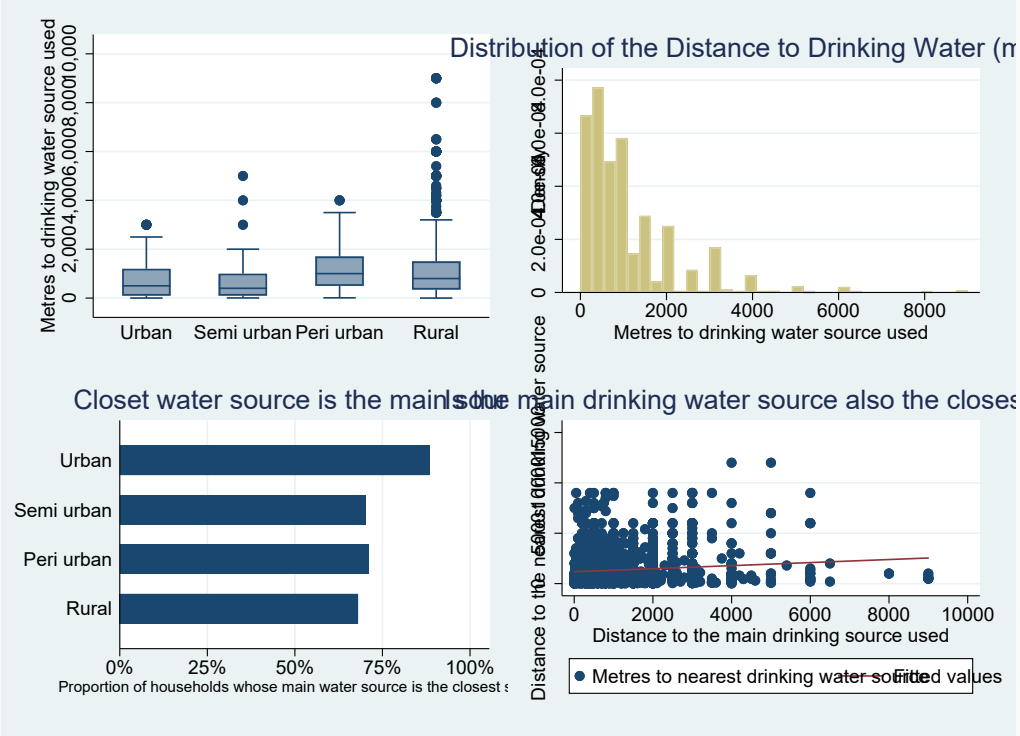


Looks ugly... and hard to understand.

Luckily, there are ways to make this nice like this.

Saving and combining graphs

Does yours look like this?



Looks ugly... and hard to understand.

Luckily, there are ways to make this nice like this.

Nonethless, let us learn how to save this in a sharable way.

Saving and combining graphs

graph save save a graph in a Stata format which cannot be opend if your audience does not have Stata.

Saving and combining graphs

graph save save a graph in a Stata format which cannot be opend if your audience does not have Stata. ***graph export*** can save in many other formats.

Its usage is very similar to ***graph save***.

Let's save this in the png format.

Saving and combining graphs

graph save save a graph in a Stata format which cannot be opend if your audience does not have Stata. ***graph export*** can save in many other formats.

Its usage is very similar to ***graph save***.

Let's save this in the png format. Your command should look somehting like this.

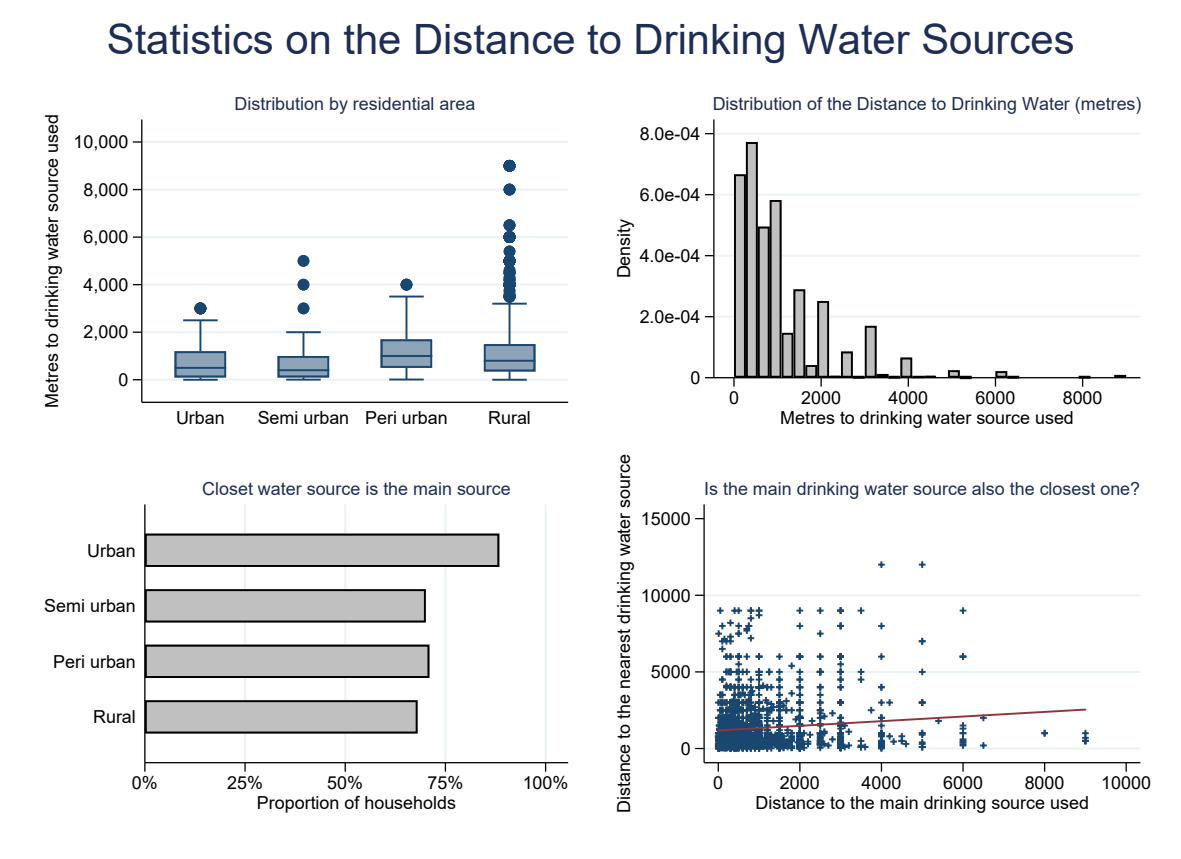
```
graph save "$script/combined.png", replace
```

Add this after your ***graph combine*** command on your dofile.

More on effective visualization

More on effective visualization

There are many ways to make it look better like the following.



More on effective visualization

Please know that it took me a while to achieve this.

You do NOT have to get everything right, but let's take some time to work on our own so we can play around with optional commands.

More on effective visualization

The box graph:

```
graph box m_drink_ws, over(urban_2012, label(labsize(small))) ///
title("Distribution by residential area", ///
          size(small)) ///
ytitle(, size(small)) ///
ylabel(, angle(0) labsize(small)) ///
graphregion(color(white))
```

More on effective visualization

The histogram graph:

```
histogram m_drink_ws, ///
fcolor(gs12) lcolor(black) lwidth(medium) ///
title("Distribution of the Distance to Drinking Water (metres)", ///
          size(small)) ///
xtitle(, size(small)) ///
ytitle(, size(small)) ///
ylabel(, angle(0) labsize(small)) ///
xlabel(, labsize(small)) ///
graphregion(color(white))
```

More on effective visualization

The bar graph:

```
graph hbar d_closest_ws , ///
over(urban_2012, label(labsize(small))) ///
bar(1, fcolor(gs12) lcolor(black) lwidth(medium)) ///
title("Closet water source is the main source", ///
          size(small)) ///
ylabel(0 "0%" .25 "25%" .5 "50%" .75 "75%" 1 "100%", ///
          labsize(small)) ///
ytitle("Proportion of households", size(small)) ///
graphregion(color(white))
```

More on effective visualization

The scatter and lfit graph:

```
scatter m_used_ws m_drink_ws, ///
        msize(small) msymbol(plus) || ///
lfit m_used_ws m_drink_ws, ///
        ylabel(, angle(0) labsize(small)) ///
legend(off) ///
xlabel(, labsize(small)) ///
ytitle("Distance to the nearest drinking water source", ///
           size(small)) ///
xtitle("Distance to the main drinking source used", ///
           size(small)) ///
title("Is the main drinking water source also the closest one?", ///
          size(small)) ///
graphregion(color(white))
```

More on effective visualization

Finally, you must make some adjustments to the combining command.

```
graph combine ///
        "$script/box_better.gph" ///
        "$script/histogram_better.gph" ///
        "$script/bar_better.gph" ///
        "$script/scatter_better.gph", ///
        graphregion(color(white)) ///
        title("Statistics on the Distance to Drinking Water Sources")
```

Importing an excel file

- You can import an excel using the drop down menu: File $\rightarrow$ Import $\rightarrow$ Excel spreadsheet
- Let us import the excel file we saved yesterday
- Use the default options in the option window. Use browse to check that the data was imported correctly
- In excel the first row can be used as variable names, but in Stata the first row is always data.
- Import the excel file using the following command. Does this add the variable names?

```
import excel using "$data\cs_s0_s5_household_modified.xls", ///
clear firstrow
```

Destringing

- Sometimes numbers are be stored as text
- We can convert numbers stored as strings to numbers by using the *destring* command
- Let us try converting the variable *earnings_sell_w* to a numeric variable

```
destring earnings_sell_w, replace
```

Some Stata do's and don't

- NEVER save over the existing dataset (especially a raw dataset)!
- Comment on why, not on what
- Always write a do-file
- Make sure your work flow works smoothly
- Provide concise and self explanatory titles to graphs

Extra section: More about Stata

Missing values

- String variables can be empty, but numeric variables can't be empty. Instead numeric variables have something called "missing values".
  - Missing values are represented in Stata with a period as in " **.** ".
  - You can also use .a or .b etc. to .z for missing values and you will learn later how these can be used
- Stata can't use missing values in computations (averages, regressions etc.) so it skips observations with missing values .
- Missing values changes the analysis as observations with missing values are excluded from commands like summarize and regress.
- Good practice to always check for missing values when tabulating variables.

Using drop down menus to create graphs

- It is hard to remember all the commands and options in general and especially to create graphs
- We can use drop down menus to create the graph
- For instance, if we want to create a pie chart for the variable *urban_2012* using dropdown menus we can follow these steps

Graphics $\rightarrow$ Pie chart, select *urban_2012* as Category variable and press OK

Defining macros - local

Type in your dofile the following and *run all the lines at once*.

- Note that ‘ is not the same as ’.

```
local numberA 3
local numberB 5
local result = (`numberA´ * `numberB´) - `numberA´
display "The result is `result´."
```

Defining macros - local

Type in your dofile the following and *run all the lines at once*.

- Note that ‘ is not the same as ’.

```
local numberA 3
local numberB 5
local result = (`numberA´ * `numberB´) - `numberA´
display "The result is `result´."
```

- What did the result say?

Defining macros - local

Type in your dofile the following and *run all the lines at once*.

- Note that ‘ is not the same as ’.

```
local numberA 3
local numberB 5
local result = (`numberA´ * `numberB´) - `numberA´
display "The result is `result´."
```

- What did the result say?

```
The result is 12.
```

Defining macros - local

Type in your dofile the following and *run all the lines at once*.

- Note that ‘ is not the same as ’.

```
local numberA 3
local numberB 5
local result = (`numberA´ * `numberB´) - `numberA´
display "The result is `result´."
```

- What did the result say?

```
The result is 12.
```

- Try running them one by one, and see what happens?

Defining macros - local

Type in your dofile the following and *run all the lines at once*.

- Note that ‘ is not the same as ’.

```
local numberA 3
local numberB 5
local result = (`numberA´ * `numberB´) - `numberA´
display "The result is `result´."
```

- What did the result say?

```
The result is 12.
```

- Try running them one by one, and see what happens?
  - It probably didn’t run. This is one of the major differences between global and local. Local is really local and only last within a single run. For more please refer to the help file on *macro*.

tabstat

- While ***summarize*** and ***tabulate*** provide useful fixed format output, ***tabstat*** gives you the ability specify exactly what statistics you want in your input.
- By default, ***tabstat*** only disply the mean.
- We can add a whole range of statistics using the option ***statistics()***. See ***help tabstat***, for a list of the statistics you can add.

tabstat

Here are some examples.

- This is the very basic command.

```
tabstat m_main_ws
```

- You can add multiple variable at a time.

```
tabstat m_main_ws m_used_ws
```

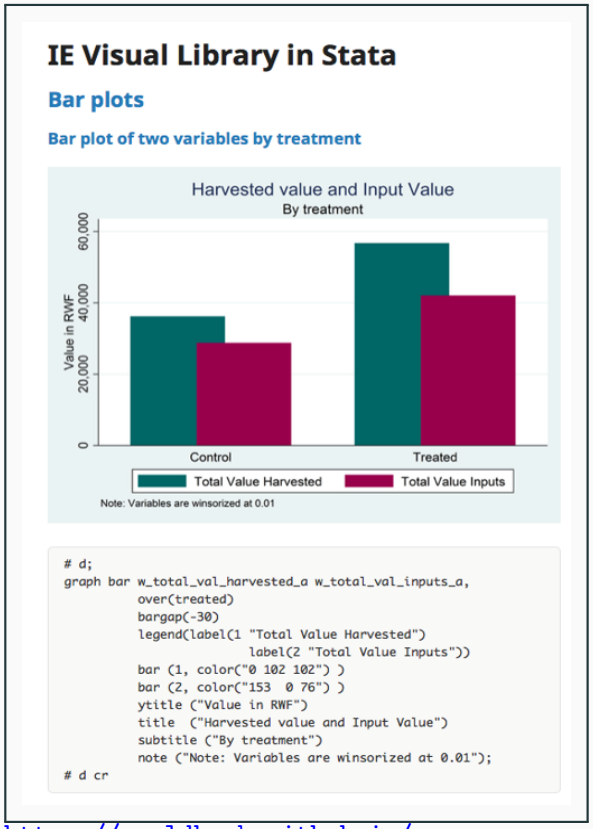
tatstat

Lastly...

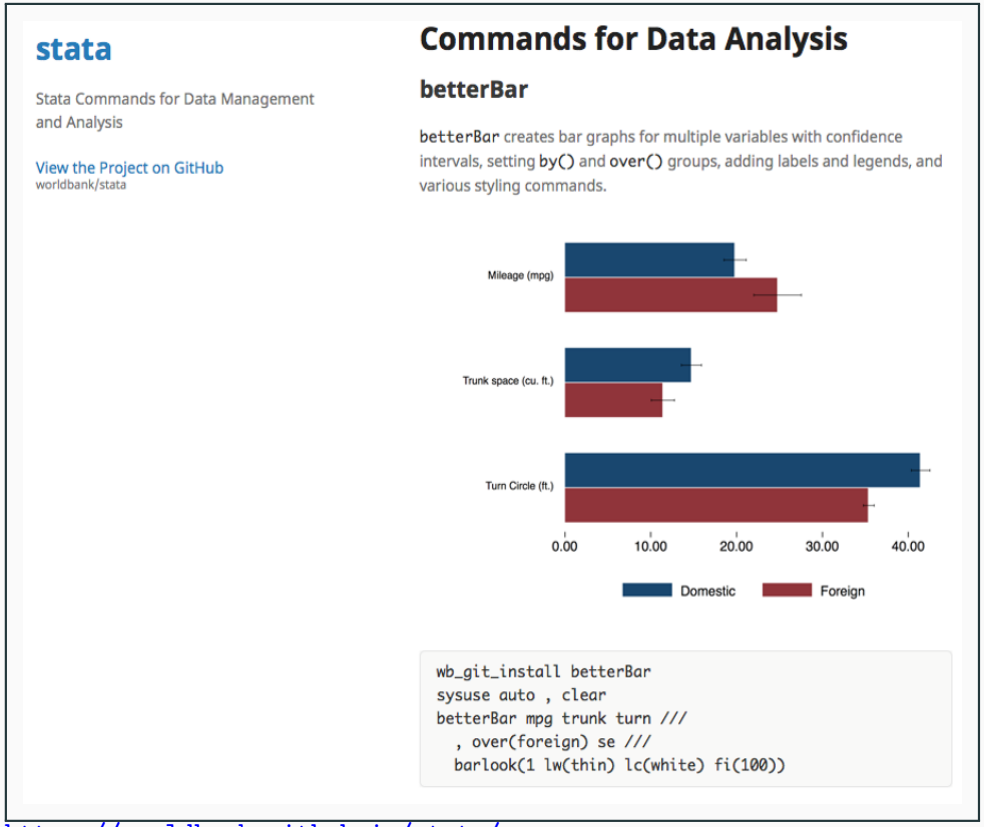
- You choose what types of statistics you want it to display.

```
tabstat m_main_ws m_used_ws, statistics(mean sd median)
```

DIME Resources



https://worldbank.github.io/Stata-IE-Visual-Library/



https://worldbank.github.io/stata/

Murakoze neza!

Sakina Shibuya and Roshni Khincha
(sshibuya@worldbank.org) (rkhincha@worldbank.org)