Development Data Boot Camp
Intro to Stata: Quick Start

Ge Sun

University of Notre Dame

May 15, 2023

Outline

Outline

Stata's interface

- Double-clicking on the Stata's icon, and you can see the interface of Stata:

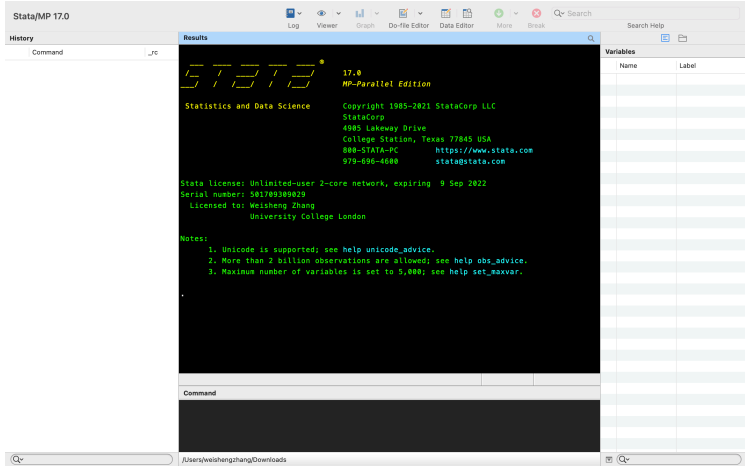
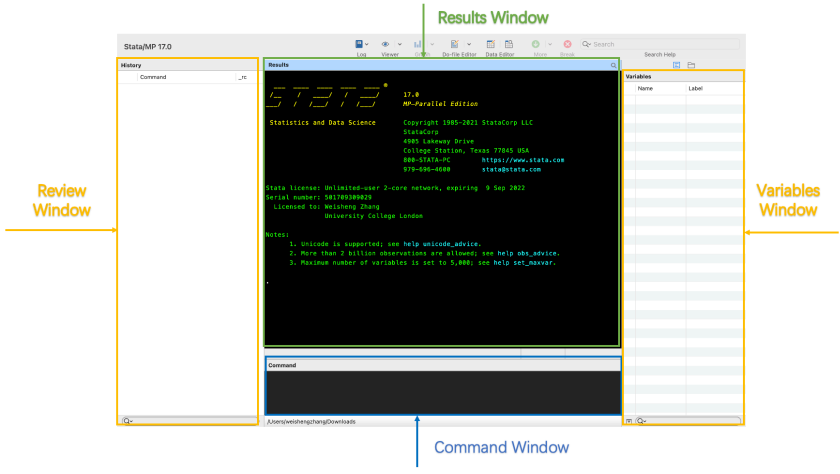


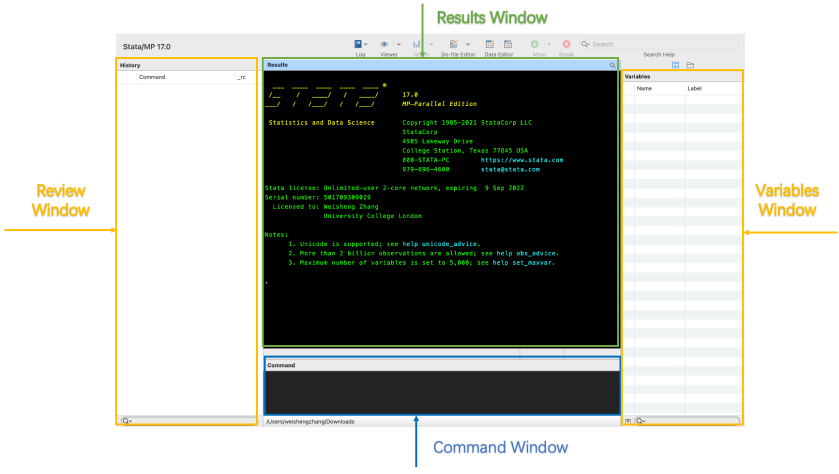
Figure 1: Stata's Interface

Four Windows



- Review window: displays previous Stata commands entered in the current session.
- Results window: displays all the commands and relative results or feedbacks.

Four Windows



- Stata command window: this is where you type commands.
- Variables window: lists all the variables in the currently open dataset

Stata's interface

A Quick Exercise

Please type the following command in the command window (command line) and then press the enter key.

```
sysuse auto.dta, clear
```

Can you tell me what happens in each of the four windows?

* *The meaning of this command:*
  Use the datasets named *auto.dta* shipped with Stata, and if there is an old dataset in the memory, clear it before importing *auto.dta*.

Stata's interface

A Quick Exercise

Please type the following command in the command window (command line) and then press the enter key.

```
sysuse auto.dta, clear
```

Can you tell me what happens in each of the four windows?

* *The meaning of this command:*
  Use the datasets named *auto.dta* shipped with Stata, and if there is an old dataset in the memory, clear it before importing *auto.dta*.

Where the data imported are?

Outline

Browse the Data

Two ways to browse the data in the memory:

- Type *browse* in the command window
- Or Click the icon *Data Editor (Browse)*.
  - Notice there are two similar icons, *Data Editor (Browse)* and *Data Editor (Edit)*. We can not edit dataset under *Data Editor (Browse)*, while we can edit the dataset under *Data Editor (Edit)*, such as change the name of variable or delete some samples.
  - *Data Editor (Browse)* is recommended as in most cases, we do not want to directly change the dataset in the data editor.

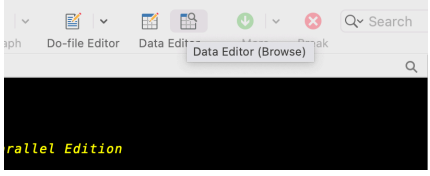


Figure 2: Click Data Editor (Browse)

Data Editor in Stata

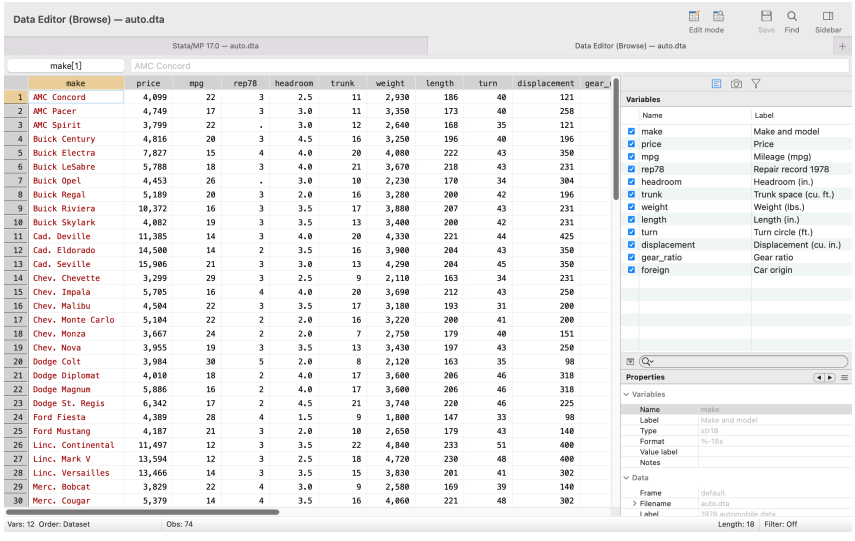


Figure 3: Data Browse

Data Structure in Stata

Data Structure in Stata:

- Column represents the variable, like the make and model of each car in this dataset, “make”, the price of cars, “price”, etc.
- Row represents the observation, containing the information of a given observation.

  * *What are the observations in this dataset?*

- Each cell contains a particular variable for a particular observation.
- Not every cell is filled with a value, some values are missing, which represented as .

Data Structure in Stata

Data Structure in Stata:

- Column represents the variable, like the make and model of each car in this dataset, “make”, the price of cars, “price”, etc.
- Row represents the observation, containing the information of a given observation.

  * *What are the observations in this dataset?*

- Each cell contains a particular variable for a particular observation.
- Not every cell is filled with a value, some values are missing, which represented as .

Quick Question:

The 48th row documents a car’s information, could you please tell me how many repair records it had in 1978?

Outline

Variable Label

- Variable label is an attribute attaching to a variable. It is very important as it provides a short description of the variable.
- You are encouraged to add a label on the important variable because it can remind you of the meaning of the variable later. It also helps other people understand your dataset.

Variable Label

- Variable label is an attribute attaching to a variable. It is very important as it provides a short description of the variable.
- You are encouraged to add a label on the important variable because it can remind you of the meaning of the variable later. It also helps other people understand your dataset.
- Two commands for label reading:

```
des [var]
codebook [var]
```

- To create label for variables:

```
label variable [var] "var_label"
```

Generate New Variables

- Command "generate" is used to generate new variables, we often use its abbreviation "gen". From example:
  - *gen new_var = 1*: generate a new variable *new_var* which value takes 1.
  - *gen wtsq = weight * weight*: generate a new variable *wtsq* which value is squared of variable *weight*.

Exercise

Generate *wtsq* variable, and label it with something that makes sense to you.

Generate New Variables

- Command "generate" is used to generate new variables, we often use its abbreviation "gen". From example:
  - *gen new_var = 1*: generate a new variable *new_var* which value takes 1.
  - *gen wtsq = weight * weight*: generate a new variable *wtsq* which value is squared of variable *weight*.

Exercise

Generate *wtsq* variable, and label it with something that makes sense to you.

- When we use *generate*, the variable generated can not be defined before. For example, if we type

$$\text{gen new\_var} = 1$$
$$\text{gen new\_var} = 2$$

there would be an error message says

variable new_var already defined

Editing Existing Variables: *replace* and *rename*

- We use command *replace* to edit the name of existing variables. From example: We have already generated a variable *new_var* which value is 1, and now we want to change the value of *new_var* from 1 to 2:

  replace new_var = 2

- Command *rename* allows us to change variable names. For example, we want to change the name for variable *wtsq* , then

  rename wtsq weight_square

  implies that *wtsq* is now called *weight_square*, and now it is legal to generated a new variable named *wtsq* by command *generate*.

Command: *drop* and *keep*

- Command *drop* allows us to remove variables or observations from the dataset in memory. For example, if we type

  drop weight_square

  the variable *weight_squre* will be removed.
- Command *keep* is like another way to drop variables because after you execute this command, only the variables that you specify will continue to be "kept".

  keep make price mpg rep78 headroom

  Also, we can keep data by the number/order of observations:

  keep in 1/5

	make	price	mpg	rep78	headroom
1	AMC Concord	4,099	22	3	2.5
2	AMC Pacer	4,749	17	3	3.0
3	AMC Spirit	3,799	22	.	3.0
4	Buick Century	4,816	20	3	4.5
5	Buick Electra	7,827	15	4	4.0

```
sysuse auto.dta, clear
keep make price mpg rep78
keep in 1/5
```

Digression but Important: Understanding Columns as the Analysis Unit in Stata

	make	price	mpg	rep78	headroom
1	AMC Concord	4,099	22	3	2.5
2	AMC Pacer	4,749	17	3	3.0
3	AMC Spirit	3,799	22	.	3.0
4	Buick Century	4,816	20	3	4.5
5	Buick Electra	7,827	15	4	4.0

Digression but Important: Understanding Columns as the Analysis Unit in Stata

	make	price	mpg	rep78	headroom
1	AMC Concord	4,099	22	3	2.5
2	AMC Pacer	4,749	17	3	3.0
3	AMC Spirit	3,799	22	.	3.0
4	Buick Century	4,816	20	3	4.5
5	Buick Electra	7,827	15	4	4.0

- The default operation in Stata is "column by column" not "cell-by-cell".
  - *This gives Stata the power to do regression analysis efficiently but also makes it not a true "programming" language because we can not handle single cell (element) as freely as we wish (Stata can not support element by element operation).*

Digression but Important: Understanding Columns as the Analysis Unit in Stata

	make	price	mpg	rep78	headroom
1	AMC Concord	4,099	22	3	2.5
2	AMC Pacer	4,749	17	3	3.0
3	AMC Spirit	3,799	22	.	3.0
4	Buick Century	4,816	20	3	4.5
5	Buick Electra	7,827	15	4	4.0

$$\text{gen mpg\_2} = \text{mpg} + 1$$

- Where would variable mpg_2 appear?

Digression but Important: Understanding Columns as the Analysis Unit in Stata

	make	price	mpg	rep78	headroom
1	AMC Concord	4,099	22	3	2.5
2	AMC Pacer	4,749	17	3	3.0
3	AMC Spirit	3,799	22	.	3.0
4	Buick Century	4,816	20	3	4.5
5	Buick Electra	7,827	15	4	4.0

$$\text{gen mpg\_2} = \text{mpg} + 1$$

- Where would variable mpg_2 appear?
- What are the values in mpg_2?

Digression but Important: Understanding Columns as the Analysis Unit in Stata

	make	price	mpg	rep78	headroom
1	AMC Concord	4,099	22	3	2.5
2	AMC Pacer	4,749	17	3	3.0
3	AMC Spirit	3,799	22	.	3.0
4	Buick Century	4,816	20	3	4.5
5	Buick Electra	7,827	15	4	4.0

replace rep78 = 0

Digression but Important: Understanding Columns as the Analysis Unit in Stata

	make	price	mpg	rep78	headroom
1	AMC Concord	4,099	22	3	2.5
2	AMC Pacer	4,749	17	3	3.0
3	AMC Spirit	3,799	22	.	3.0
4	Buick Century	4,816	20	3	4.5
5	Buick Electra	7,827	15	4	4.0

replace rep78 = 0

drop rep78

Digression but Important: Understanding Columns as the Analysis Unit in Stata

	make	price	mpg	rep78	headroom
1	AMC Concord	4,099	22	3	2.5
2	AMC Pacer	4,749	17	3	3.0
3	AMC Spirit	3,799	22	.	3.0
4	Buick Century	4,816	20	3	4.5
5	Buick Electra	7,827	15	4	4.0

replace rep78 = 0

drop rep78

keep make

Digression but Important: Understanding Columns as the Analysis Unit in Stata

	make	price	mpg	rep78	headroom
1	AMC Concord	4,099	22	3	2.5
2	AMC Pacer	4,749	17	3	3.0
3	AMC Spirit	3,799	22	.	3.0
4	Buick Century	4,816	20	3	4.5
5	Buick Electra	7,827	15	4	4.0

replace rep78 = 0

drop rep78

keep make

egen mean_mpg = mean(mpg)

Digression but Important: Understanding Columns as the Analysis Unit in Stata

	make	price	mpg	rep78	headroom
1	AMC Concord	4,099	22	3	2.5
2	AMC Pacer	4,749	17	3	3.0
3	AMC Spirit	3,799	22	.	3.0
4	Buick Century	4,816	20	3	4.5
5	Buick Electra	7,827	15	4	4.0

replace rep78 = 0

drop rep78

keep make

egen mean_mpg = mean(mpg)

egen mean_rep78 = mean(rep78)

Command: help

- *help* command displays detailed information about the specified. For example, type

  help rename

  and there will be a window displaying the document of how to use *rename*.
- Sometimes we do not remember the name of the Stata command, then we can use command *search* by keyword. For example, type

  search label

  and Stata will display a list of commands and resources involving keyword “label”.

Basic Stata Grammar

The basic structure of grammar is:

[prefix] : command [varlist] [= exp] [if] [in] [weight] [using filename], [options]

- *prefix* denotes a command that repeats execution of *command* or modifies the input or output of *command*.

bysort county: egen average_income_country = mean(income)

- *command* denotes a Stata command,
- *varlist* denotes a list of variable names,
- *exp* is a mathmatical expression,
- *weight* denotes a weighting expression,
- *filename* is a filename, and
- *options* denotes one or more options that apply to *command*.

Outline

Importing Data from External Sources

- Stata can read other types of data from external sources including:
  - Excel spreadsheet (*.xls, *.xlsx)
  - text data (*.csv)
  - Stata data(*.dta)
  - DCT file (*.dct)
- Stata has various commands for importing different kinds of data from external databases:
  - The *import excel* command can read Microsoft Excel files.
  - The *import delimited* or *insheet using* command can read *.csv files.
  - The *use* command can read *.dta files. (sysuse auto.dta)
  - The *infile* command can read *.dct file.

Importing data from external sources

The basic syntax of importing external data is:

import file_type using file_name, specific_options_wrt_file_type

- Replace *file_type* with the type of your file, like *excel* or *spss*.
- Replace *file_name* with the path and the name of your file, like */Users/sybil/Documents/school_gps_map.csv*.
- There are various options with respect to your type of file. For example, some widely used options for Excel spreadsheet are:
  - *firstrow*: treat first row of Excel data as variable names.
  - *sheet("sheetname")*: decide which Excel worksheet will be imported into Stata.
  - *clear*: replace data currently used in Stata (memory).

Importing data from external sources

- The syntax of importing external data into Stata can be complex and hard to remember. Fortunately, we do not need to remember it with the help of GUI.
- As a beginner, we can use the following steps: (1) Import the data using the GUI, and (2) Copy the code that Stata generates automatically and paste it into your do-file, so that the process is reproducible.

Importing data from external sources.

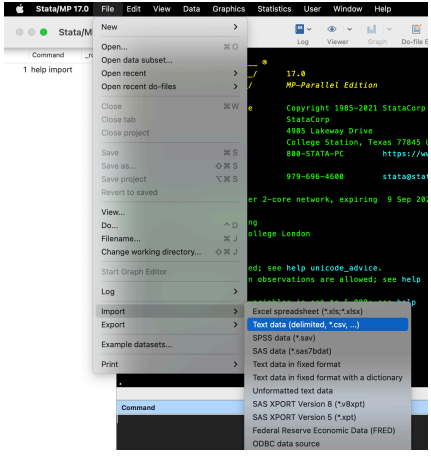


Figure 4: Choose the corresponding command

Importing data from external sources.

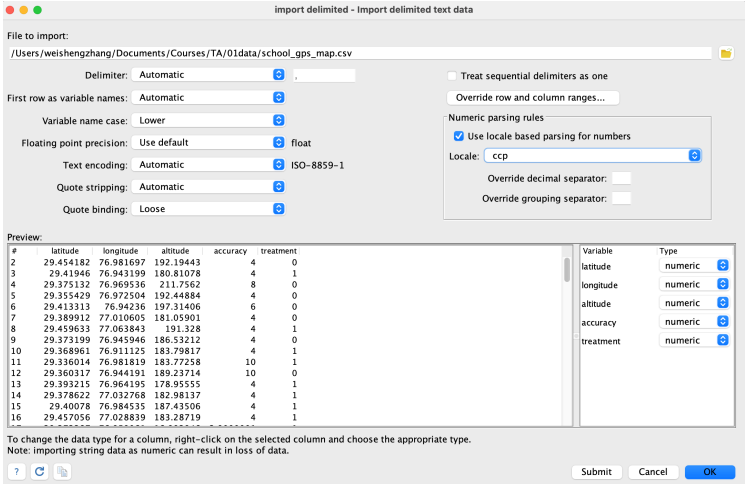


Figure 5: Choose the file and options

Outline

Types of Data

- Every variable in Stata is either type **numeric** or type **string**
- Numeric:
  - Numeric type refers to variables that are stored as numbers.
  - You can use $>$ and $<$ to compare them.
  - You can enter them into arithmetic calculations.
  - You can get the mean, the quantiles for numeric variables.
- String:
  - String type refers to variables that are stored as words/sentences.
  - It is easy to extract different pieces of information from one sentence.
  - There are some magic commands that will help us achieve that goal.

Numeric type data

What can numeric type data be?

- **Int** Variables that are integers.
  - For example: year, age, and number of siblings.
- **Float (or double)** Variables that can take continuous values.
  - For example: income, consumption, and assets.
- **Byte** Variables with type byte are categories, i.e., variables that can only have finite values.
  - dummy variables (gender, whether having an insurance, whether living in rural areas).
  - education level (primary school, high school, college, etc.)
  - race (black, white, Asian, Hispanic, etc.)

In most cases we do not need to pay much attention to the difference between int and byte, float and double.

String type data

What can string type data be?

- country (USA, UK, etc.), state (Indiana, California, etc.)
- date (like 1994/01/01, which is a string instead of a number, we will deal with that later)
- Frequently used string types are:
  * strX, where X maximum length of the string. For examples, str5 refers to string variables with maximum length 5.
  * strL. strL indicates that variables can hold strings of arbitrary lengths.

In most cases we do not need to distinguish str3, str5, and strL.

Extract useful information: split

- Sometimes we need to contract useful information from string variables.
- Consider variable date in the format 1940/12/20, we want to contract information in this variable and attribute it into three variables, year, month, and day.
- Command *split* can help us. The usage of is

```
split date, gen(new_var), parse(/)
```

- Command *regexm* is more powerful, you can use help document to explore it whenever you need it.

Convert one type to the other

- Sometimes we need to convert string variables to numeric variables. It often happens when we import excel or csv data into Stata, as in this case, Stata treat every variable as strings, even the variable only contains numbers.
- Command *destring* can help us. The usage of *destring* is

```
destring var, replace
```

or

```
destring var, generate(new_var)
```

Convert one type to the other

- Sometimes we need to convert string variables to numeric variables. It often happens when we import excel or csv data into Stata, as in this case, Stata treat every variable as strings, even the variable only contains numbers.
- Command *destring* can help us. The usage of *destring* is

```
destring var, replace
```

or

```
destring var, generate(new_var)
```

- Numeric to string: *tostring*

Outline

Commands: describe and codebook

- Two commonly used commands to explore the dataset:
  - *describe*. Provide an overview of the dataset:
    - Number of observations in the dataset.
    - Number of variables in the dataset.
    - Other basic information about the variables in our dataset, like types, labels, etc.
  - *codebook*. Provide more detailed information of variables in the dataset, like range and mean.
- Commands *describe* and *codebook* can also be used as exploring specific variables.

Commands: lookfor and count

- *lookfor* helps you find variables by searching for string among all variable names and labels.
  - It helps find the relevant information you need, especially in a large dataset with tons of variables.
- *count* displays the number of observations in the data.
  - You can add conditions to ask for the number of observations that satisfy the specified conditions.

    count if rep78 $>$ 3
  - It is very useful in debugging the codes!

Commands: tabulate and summarize

- *tabulate* is an extremely useful command in exploring data.
  - When only one variable is specified, i.e., *tab var_name*, *tabulate* gives count of times and percentage of each value appears in data.
  - When two variable is specified, i.e., *tab var_name_1, var_name_2*, *tabulate* gives the relationship between these two variables, where var_name_1 appears as row, var_name_2 appears as column.
- *summarize* is useful when we want to know basic summary statistics of some variables, i.e., mean, standard error, and median.
- *tabulate* might not work if there are too many distinct values of the variable, and it is often applied on category variables, like gender and occupation. *summarize* is often applied on continuous variables, such as incomes, as it is meaningless to calculate the mean of people's occupations.

Commands: histogram

- When we are faced with a continuous variable, it is more convenient to look at a graph than reading tedious numbers.
- *hist* is a useful command to help us understand the big picture of continuous variables, it gives us a histogram of the distribution of a specific variable.
- Graph often tells us more than summary statistics and reveals some subtle problems. My recommendation is, always make histograms of critical variables in your research.

Missing Values

- A missing value is . in numeric format, and it is "" in string format.
- We can use

$$\text{gen miss\_var\_name} = \text{missing(var\_name)}$$

  to take on the value 1 for true if its argument is missing and the value 0 for false, otherwise. It can be abbreviated mi().
- Use *tab var_name, m* to explore how many observations have missing *var_name*. Command tabulate will omit missing values without option *m*.
- Regression analysis will omit all observations having a missing value for any one of the variables used in the model. (specification 1 v.s. specification 2)

Outline

Where do you work

- **Directory**: A directory is a unique type of file that contains only the information needed to access files or other directories. (what is the folder you are working on)
  - Question: if you save some outputs in Stata now, where are you going to find them?

Where do you work

- **Directory**: A directory is a unique type of file that contains only the information needed to access files or other directories. (what is the folder you are working on)
  - Question: if you save some outputs in Stata now, where are you going to find them?
- Showing the current directory:

  pwd (short for "print working directory")

Where do you work

- **Directory**: A directory is a unique type of file that contains only the information needed to access files or other directories. (what is the folder you are working on)
  - Question: if you save some outputs in Stata now, where are you going to find them?
- Showing the current directory:

  pwd (short for "print working directory")
- Change the directory and save the data there:

  cd "/Users/gesun/Desktop/Bootcamp2023/02_Intro_to_Stata/Class_Exercise"

  save auto_exercise.dta, replace

Where to find your work

Path: an address or location to help your computer locate the file you are searching for. (it is like the mail address to send you a postcard)

- Absolute path: a location in a file system relative to the root directory
  - Root directory: the top-level directory in a folder structure.
    - C:\ for Windows
    - / for Mac
- Relative path: a location in a file system relative to the current directory you are working on

Where to find your work

Example

```
cd "/Users/gesun/Desktop/Bootcamp2023/02_Intro_to_Stata/Class_Exercise"
use auto_exercise.dta,clear
```

Where to find your work

Example

```
cd "/Users/gesun/Desktop/Bootcamp2023/02_Intro_to_Stata/Class_Exercise"
use auto_exercise.dta,clear

cd "/Users/gesun/Desktop/Bootcamp2023/02_Intro_to_Stata"
use "./Class_Exercise/auto_exercise.dta", clear
```

- "./" meaning: sub-folder
- *for Window users, replace "/" with "\"*

Where to find your work

- Another reason why fold structure is important

```
cd “your_project_folder”
use “./01_raw_data/data_file_name.dta”,clear
do “./02_do_files/clean_data.do”
save “./03_temp/cleaned_data.dta”,replace
```