Loading Data and an Introduction to Variables and Labels

Stata Data

- When Stata begins, no data is loaded into memory, and the Variables, Data Editor, and Data Browser windows will all be empty

- Stata works with one dataset at a time. In order to examine other datasets the user must explicitly switch between them

- Stata has built-in protections to help users avoid accidentally **`clear`**-ing unsaved data from memory or **`save`**-ing over files on the hard drive

Loading Data

- The most streamlined way of loading data into Stata is by loading a Stata `.dta` file

- In this case, you simply use the command:

```
use filename
```

- Stata comes with toy datasets already loaded for teaching purposes, as well as easily accessible web-based datasets

```
sysuse dataset
```

```
webuse dataset
```

Excel Files

- One of the many positive features of Stata is its strong Excel file support (**.xls** and **.xlsx** files)

```
import excel filename, firstrow
```

- This command will load the Excel file into Stata and use the first row to determine the variable names

- When possible, use the import excel command because it correctly deals with dates and number types

    - Without the `firstrow` option, the variable names will be set as Excel column letters (e.g. A, B, C, D)

Delimited Files

- Stata can easily import comma-separated (**.csv**) files or tab-separated (usually **.txt**) files using one command:

```
import delimited filename
```

- The import command can specify only certain rows or columns as well

```
import delimited filename, rowrange(start:end)
colrange(start:end)
```

- This is one of many options available. Use `help import` to browse through them all!

Fixed Format Files

- Stata can also import fixed-format files using dictionary files
    - These dictionary files must define the types for each column

```
infile using dictionary_file, using(data_file)
```

- For examples of this type of file loading, see **`help`** `infile fixed` (not a file type I am very familiar with)

Navigating Directories

- When loading any file, you need to make sure that Stata is pointing to the correct directory (where that file is present)
    - It is possible that your Stata session is currently "pointing" to a different folder on your computer
- To check where you currently are, use the command **`pwd`**
- To take a look at what is in your current directory you can use the **`ls`** command
- In order to change directories, you can use the **`cd`** command along with a folder in your current directory or a filepath

Variables

- The word “variable” can mean many things in programming and statistics, but in Stata it refers particularly to the social scientist’s common definition of a variable
    - In databases this might be called a *field,* or *column*. When using the Data Browser or issuing the command `list`, variables will be represented as *columns*
    - There are other “variables” held in memory and available for Stata users, but these remain hidden to the beginner

Observations

- In Stata, observations refer to the entries in your dataset, each of which has values for some, if not all, of your variables

- An observation is usually considered the unit of your analysis. Often, each observation is unique in some way, but for certain analyses complete duplicates may be present

- Stata observations make up the *rows* of your dataset when calling `list` or using the Data Browser

Basic Commands

- In order to learn some basic things about variables, we will be using some basic commands:

- Commands are actions performed on datasets

`generate` — generates a new variable

`replace` — replaces an already existing variable

`describe` — describes the details of a dataset or variable

`drop` — drops variables or observations (when combined with an if statement)

`keep` — keeps variables or observations (when combined with an if statement), the opposite of **`drop`**

generate

```
generate variable_name = expression
```

- This command generates a new variable based on a given expression
- The expression can be a constant (e.g. a number 4, or a string “Hello!”) or can be dependent on other variables (e.g. `weight/height` or `age + 5`).
- This command will only generate a new variable if that variable name is available

replace

```
replace variable_name = expression
```

- This command replaces the values of a pre-existing variable based on a given expression

- Just like **generate** this can be a constant or dependent on other variables (e.g. mpg*price).

describe

describe

describe variable_name

- This command will output a description of the variable, including the variable name, storage type, value label, and variable label

```
. describe rep78

              storage   display    value
variable name   type    format     label      variable label
------------------------------------------------------------------------------
rep78           int     %8.0g                 Repair Record 1978
```

drop

```
drop variable_name
```

- The **drop** command with `variable_names` will remove ***variable***(s) from the dataset

```
drop if rule
```

- The **drop** command with a `rule` will drop ***observations*** that satisfy that rule from the dataset

keep

- The **keep** command is the reverse of the **drop** command

```
keep variable_name
```

- The keep command with variable_names will keep only the ***variable***(s) specified in the dataset

```
keep if rule
```

- The keep command with a rule will keep only ***observations*** that satisfy that rule in the dataset

`if` rules

- This `if` is the `if` qualifier, not the `if` from a `if/else` programming statement
- `if` qualifies a command, telling Stata that we want to perform our action on only a subset of our data
- In the case of **`drop`** or **`keep`** commands, `if` tells Stata what observations to drop, or what observations to keep

```
keep if mpg > 25

drop if headroom < 2
```

if rules

```
keep if mpg > 25
```

The rule evaluates to a True (1) or False (0) for each observation. If it evaluates to True, the command is performed on that observation.

```
keep if headroom == 2
```

Rules that check equality use double equal signs. This == tells Stata you are checking equality, not assigning something.

if rules

```
keep if mpg >= 25 & headroom < 3
```

(mpg is greater or equal to 25 AND headroom is greater than 3)

```
keep if headroom != 2 | mpg > 30 | price < 80000
```

(headroom does not equal 2 or mpg is greater than 30 or price is less than 8000)

Rules can use boolean logic operators like *AND* and *OR.* In Stata the ampersand **&** represents AND, while the vertical line | represents OR. The entire statement will be evaluated as True (1) or False (0).

Rules that check equality use double equal signs. This == tells Stata you are **checking for equality**, *not assigning something*. The != symbol combination means **not equal**.

Labels

- Labels are a built-in way of conveying metadata about your variables and dataset to others and to yourself
- There are several labeling scopes in Stata (dataset, variable, variable levels (i.e. values))
- For now, we will take a look at dataset and variable labels

Dataset and Variable Labels

label data *label*

- This command will label your entire dataset with a particular *label* that can be seen when using the command **describe dataset_name**

label variable **variable_name** *label*

- This command will assign a *label* to a variable in your dataset that can be seen when using the command **describe variable_name** or by looking in the variables portion of your Stata window

Exercises (1)

1. Titanic Data

    A. Create a do file called titanic.do

    B. Load the titanic.csv file into Stata

    C. Drop all variables except name, age, sex, survived, and fare

    D. Only keep observations whose age is greater than 20 and who survived

    E. Try your best to give informative labels to each variable

Exercises (2)

2. Movie Metadata

   A. Create a do file called movies.do

   B. Load the movie_metadata.xls into Stata (remember the **firstrow** option)

   C. Drop all movies with runtimes equal to or less than 45 minutes

   D. Keep the variables: duration, gross, movie_title, country, budget, and imdb_score

   E. Limit the dataset to only movies from the United States