# Stata Training Day 2: Creating Usable Datasets

## Importing

You can import several file types into Stata. Most common among them are comma separated values (CSV), Excel worksheets (XLS/XLSX), and Stata’s own DTA files. We call these imports “raw data” -- i.e. data we haven’t manipulated yet since it comes straight from the source.

### Comma separated values

**import delim**ited “acs\_17\_pop\_stats.csv”, varnames(#) clear

### Excel worksheets

import excel using “lending\_groups.xlsx”, firstrow clear

### Stata files

use “acsmarriage.dta”, clear

## Loops

### Foreach

Foreach loops allow you to loop over items and handle more repetitive tasks.

How can we use a foreach loop to generate new variables with information on the actual number of married, separated, widowed, and never-married people in each state, rather than a percentage?

foreach var of varlist sep married widow never {

gen `var’\_raw = `var’\_perc \* total\_pop /100

}

### Forvalues

Forvalues allows you to loop over consecutive values. Let’s take our example from the American Community Survey. We have information on the percentage of the population who are married by state.

How can we use a forvalues loop to display the number of married people from first 10 observations?

forvalues x = 1/10 {  
 di married\_raw[`x’]

}

## Merging

### merge 1:1

### merge m:1

### merge 1:m

### mmerge

mmerge is a user-created command that executes m:1 and 1:m merges safely. The structure of mmerge is very similar to Stata’s own merge commands:

mmerge countrycode using “otherdata.dta”

Because it is user-created, you must install a version before using it. You can do this by typing **ssc install mmerge** into the command line or by including it in your do file.

### 

## Manipulating Variables

### Conditions & Syntax

#### If

Across coding languages, **if** is a conditional statement used to qualify the command before or after it. Let’s say we wanted to see which states have a total population greater than 20 million.

**tab**ulate state if total\_pop > 20000000

We see California and Texas listed because they are the only two states with a population over 20 million. What if we wanted to see which states have a total population under 500,000?

**tab**ulate state if total\_pop < 500000

Our answer is Wyoming, where the total population is 426,614. How would you go about tabulating the total population of Wyoming? Use the if condition!

#### And (&), Or (|), and Not (!=)

Sometimes adding one condition is not enough. We use the ampersand (&) and vertical line (|) symbols to apply and/or conditions, respectively, to our data in Stata.

Using the example above, let’s add another aspect to our conditional statement. We want to see which states have a total population under 2 million and a divorce rate above 12%.

**tab**ulate state if total\_pop < 2000000 & divorce\_perc > 12

Here we find that there are 9 states where the divorce rate is over 12% and the total population is under 2 million. Maybe we decide we want to know which states have a total population under 2 million OR a divorce rate over 12%, which would look like this:

**tab**ulate state if total\_pop < 2000000 | divorce\_perc > 12

Lastly, understanding that our data contains U.S. territories and our capital as observations, how can we display only states? The “not” syntax (!=) can help us out.

**tab**ulate state if state != “Puerto Rico” & state!=”District of Columbia”

### Generating Variables

Suppose that it’s necessary to analyze our data by region in addition to by state. Except we only have states and their respective statistics. How can we create a new variable that represents the four regions of the United States? We first start with the “East” but be careful: only a select few states can be categorized as “East” so we need to use the condition and syntax we learned before.

**gen**erate region = “East” if state == “Connecticut” | state == “Maine“ | state == “Massachusetts“ | state == “New Hampshire“ | state == “Rhode Island” | state == “Vermont“ | state == “New Jersey“ | state == “New York“ | state == “Pennsylvania“

### Replacing Values

Notice how in the above example, we generated a conditional variable that only contains values for any state located in the “East” region of the United States. How can we modify the region variable so that it contains values for all the states in our dataset? Using the replace command, along with another conditional statement, we can add three more regions and assign them to the remaining states.

replace region = “Midwest” if state == “Illinois” | state == “Indiana” | state == “Michigan” | state == “Ohio” | state == “Wisconsin” | state == “Iowa” | state == “Kansas” | state == “Minnesota” | state == “Missouri” | state == “Nebraska” | state == “North Dakota” | state == “South Dakota”

replace region = “West” if state == “Arizona” | state == “Colorado” | state == “Idaho” | state == “Montana” | state == “Nevada” | state == “New Mexico” | state == “Utah” | state == “Wyoming” | state == “Alaska” | state == “California” | state == “Hawaii” | state == “Oregon” | state == “Washington”

replace region = “South” if state == “Delaware” | state == “Florida” | state == “Georgia” | state == “Maryland” | state == “North Carolina” | state == “South Carolina” | state == “Virginia” | state == “District of Columbia” | state == “West Virginia” | state == “Alabama” | state == “Kentucky” | state == “Mississippi” | state == “Tennessee” | state == “Arkansas” | state == “Louisiana” | state == “Oklahoma” | state == “Texas”

### Renaming Existing Variables

Renaming variables is one of the simplest form of data manipulation in Stata.

rename state estado

We can also rename multiple variables at the same time by putting the existing names in parenthesis and the new names in the following parenthesis.

rename (total\_pop id2) (pop id)

### Encoding / Recoding

encode region, generate(reg\_code)

recode reg\_code (1=2) (2=3) (3=4) (4=1)

### Drop or Keep

Unfortunately, not all variables or observations are worth keeping in the dataset. When we merged the ACS marriage statistics file with its total population counterpart, we ended up with a variable called \_merge. This is a helpful variable to browse through if your merge didn’t go as planned but for this exercise, all our observations matched. In that case, we can simply type:

drop \_merge

But what if we have a very large dataset and only want to keep a handful of variables? This is where the keep command comes in handy. If we have a large dataset and only want to retain mar\_perc q10 and id, we would type:

keep mar\_perc q10 id

This same logic works for observations, provided we give conditions. Say we want to drop all observations in the dataset that have a missing id. We can do the following:

drop if missing(id)

### Labeling Variables or Values

**la**bel **var**iable region “Region”

**la**bel **def**ine region 1 “Northeast” 2 “Midwest” 3 “South” 4 “West”

**la**bel **val**ue reg\_code region

## Reshaping