Lab 2: Basics of Stata

Simon Halliday

ECO311, Fall 2016

Revision Quiz

In your notes for your exercises for this lab, try to answer the following quiz questions based on Lab 1 *before class* and *without checking the lab*! In your exercises, have three categories for each question, “M" for my answer, “TPS” for think-pair-share, and “N" for after notes.

a. What command did we use to find a frequency table for a variable?

b. What is it important for you to remember to do when you use the command `log using` to open a log file to track your work?

c. What command opens up a spreadsheet that helps you to look at your data? What would you do to look at the spreadsheet for only one observation?

d. What does the command `codebook` do?

e. What does the logical operator $!=$ mean?

f. What would the logical operator be for two conditions to hold at the same time? e.g. for someone to be a working age female.

g. When I specify *when* a command should be run using the command `if`, do I need to put a comma before the command (that is, is `if` an “option” for commands)?

When you get to class, check your answers with your neighbors. Once you have checked with your neighbors, go back to the lab to check what you didn’t remember from the lab notes.

Doing Stata
By the end of this lab, you should have:

- Replace values of variables, e.g. `replace`
- Explore the data and their summary statistics using basic Stata commands, e.g. `tab`, `sum` and `tab, sum()`
- Describe the difference between continuous and discrete variables
- Describe the difference between individual-level and household-level variables
- Create new variables from old variables, i.e. `generate` and `egen`
- Identify one observation to represent information for the household
- Examine subsets of your data
- Summarize data using scatter plots and line graphs, `graph twoway` ...
- Saving your data, `save using`

Introduction

Before you start this lab session remember to do the following:

- Direct your working directory to the Labs folder (or Lab2 folder)
- Make sure that your file structure is consistent with Lab1 (CommandFiles, log, cmdlog, etc).
- Open a `log` file to track your work:

```
log using .\ProcessingAnalysis\CommandFiles\logs\
    yourlastname_lab2.log
```

Be sure to specify the file extension `.log`

- Open a cmdlog file

  ```
  cmdlog using .\ProcessingAnalysis\CommandFiles\cmdlogs\
      yourlastname_lab2.do
  ```

After you have followed these steps, copy the NIDS data into your Original-Data folder for Lab2 and then open the data.

Re-coding variables

As we saw in Lab 1, sometimes the data arrives and it is in a form that we cannot immediately use or interpret.

For example, missing data are often coded with negative numbers or large numbers with the numerals 9999. However, if someone for years of education had the code 9999, they certainly would not have 9999 years of education! We need to re-code the 9999 to be a period “." which Stata uses to read a variable as “missing.”

Another example is that gender is often reported with one of the genders, say Male, being coded as a 1 and female being coded as a 2. But, if we want to use gender as a dummy variable in regression analysis, then we need to re-code this gender variable so that female equals zero.

How do we do this in Stata? We use the `replace` command. Let’s look at an example now.

Re-coding Education in NIDS

In Stata a missing value is usually recorded as a full stop/period. NIDS uses -9 (or 99, 9999 etc) for “Don’t know,” -8 for “Refused,” -5 for “Not applicable” and -3 for “Missing.” When we start to analyze the data we will need to re-code these negative numbers to missing so that Stata knows to treat these observations as missing.

So, let’s use the `replace` command:

```
replace w1_r_b7 = . if w1_r_b7 < 0
```

Notice that we’ve used the condition `if w1_r_b7 < 0`. What that tells Stata to do with the `replace` command is to find all of the rows (observations) for which the value of the variable `w_r_b7` are less than zero and to replace them with a period (i.e. set them equal to “.” so that Stata reads them as “missing”).

What can we do with the variable `if w1_r_b7` now that we’ve re-coded it? Let’s try a cross-tab of education and race:

```
tab w1_r_b7 w1_best_race
```

Now tab *excludes* all of the missing data when making these calculations.

You can *include* the missing data by stipulating `, missing` as an option.

```
tab  w1_r_b7 w1_best_race, missing
```

When we summarize variables and when we start to analyze relationships between variables, we need to be very careful about missing data for numeric variables – the period can be interpreted as positive infinity by Stata in certain situations.

Basic Summary Statistics

In Lab 1 we looked at how to use the `tab` command (for tabulate) to look at frequency tables and cross-tabs of different variables. But, we often want to know more than just the frequency of different variables. For example, we may want to know the median, mode, or values at different percentiles in the data, e.g. the value at the 90th percentile. We can start to get such information from the `summarize` command, which can be abbreviated to `sum`.

But remember to use your statistical understanding to know when it is appropriate to look for summary statistics vs. frequency tables.

Consider the following two commands:

1. `sum w1_r_b6`
2. `sum w1_best_race`

Having run those two commands, complete the exercise.

Exercise 1

a. What does the output from the first command tell you?

b. What does the output from the second command tell you?

In answering your questions, consider the difference between a *categorical* or *discrete* variable and a numerical variable.

When you ran the second command what was the *minimum* value? Does that value make sense for the variable it represents? Explain and add to your explanation in the exercise. Can you re-code the variable `w1_r_b6` in the same way we did with variable `w1_r_b7` so that it makes sense? What value do you find for the variable once you've re-coded it? How old is the youngest person? Does the minimum make sense in terms of how the variable is defined?

HINT You should find that the average *before* you re-code it is 26.5 and the average *after* you re-code it is 27.4. You can find a solution with code at the end of the lab.

Sum and logical operators

The sum command may also be used with any of the qualifying, comparison or logical operators (check the table of logical operators from Lab 1 if you need a refresher). `sum` may be used to generate sample statistics for multiple variables at a time. Think about what each of the following commands will give you, and then execute them (label this as Exercise 2 – write what you think the commands mean in English *before* you run the command, then add notes *after* you've run the command to check it). Look up the variables before you run each command.

```
sum w1_r_b6 w1_r_b7 w1_hhincome
sum w1_r_b6 if w1_r_b4 == 2
sum w1_r_b7 if w1_r_b6 >= 18
sum w1_r_b7 if w1_r_b6 >= 18 & w1_r_b6 != .
```

Look at the number of observations for the last two commands. Notice how STATA regards missing values as positive infinity – we need to be extremely careful about missing values. The `summarize, detail` command will give even more detailed statistics such as the value of the variable at each of a number of percentiles, the variance, skewness and kurtosis of the data. For example,

```
summarize w1_r_b6, detail
```

tells us that 75% of the sample have age values less than 41 years, the skewness of the age data is 0.748 and the kurtosis is 2.82.

SKEWNESS & KURTOSIS Skewness tells us how asymmetrically distributed the values of a variable are. If the skewness measure is zero, it means the values in the variable are *symmetrically* distributed about the mean. If the skewness coefficient is > 0 (< 0), then the values are skewed to the right (left) of the mean. Kurtosis tells us about the *thickness* of the tails of the distribution, or what proportion of our sample has values for the variable that lie in the tails (end points) of the distribution.

The tab, sum command

The tabulate and summarize commands may also be used together. Used together, they create a frequency table from the variable indicated with the use of the `tab` command along with the summary statistics created by the use of the `sum` command.

For example, we might want to know the average age of each population group in the sample, or the average age of each sex, or the average monthly household income for each population group. For these questions, we would type:

- `tab w1_best_race, sum(w1_r_b6)`
- `tab w1_r_b4, sum(w1_r_b6)`
- `tab w1_best_race, sum(w1_hhincome)`

We can also use the tabulate and sum in a two-way table. For example we might want to know the average age by population group and sex. Here we would type:

```
tab w1_best_race w1_r_b4, sum(w1_r_b6)
```

It can be seen from the output that the table gives us means, standard deviations and frequencies for each cell. We could feel overwhelmed by all the information. If we are interested only in means, we can request this by typing:

```
tab w1_best_race w1_r_b4, sum(w1_r_b6) means
```

If we want means and standard deviations but not frequencies, we type:

```
tab w1_best_race w1_r_b4, sum(w1_r_b6) means standard
```

Or,

```
tab w1_best_race w1_r_b4, sum(w1_r_b6) nofreq
```

Tabsum is a key tool in descriptive analysis. There are more options than we show here. Use `help tabsum` in Stata to find out about these options.

Exercise 2

1. Provide a command that generates a table that displays in each cell the mean household expenditure by population group and sex.
2. What is the mean household expenditure for Black women?
3. What is the mean household expenditure for white men?

Continuous vs. categorical variables

In the NIDS data set there are several types of variables. Economists, sociologists, and psychologists use different language to describe the multiple types of variables. Here, we will divide variables into two types: *continuous* variables and *categorical* variables. The statistical and graphical tools we use to understand the distributions of the various types of variables differ greatly, so we need to understand the differences between these measures.

Continuous variables

Continuous variables have an infinite number of possible values that fall between any two observed values. For example, consider age. In our data, age is recorded in years. But it could have been recorded in months, days, minutes, or even seconds. A continuous variable is *ordinal* in the sense that its values have an inherent order. In the age example, an age of 16 years is one year older than the age of 15 years, so the unit of measurement in between these two values is itself meaningful. (This may seem like common sense, but when we consider categorical variables, this will no longer be true). Examples of continuous variables in the NIDS data set include age (`w1_r_b6`) and household income (`w1_hhincome`).

We are actually not being terribly careful with our definitions. Consider, for example, a variable that counts members of the household (`w1_hhsizer`). Household size might be 4 or 5, but it will never be 4.34. Nonetheless, if we were told that the average number of members of a household was 4.34, it would be comprehensible. We would know that, on average, there are more than 4 members and less than 5 members in the households in the dataset. We are going to treat variables like household size and number of births as continuous variables: taking the average gives an answer that is readily interpreted. Taking the average of a categorical variable, on the other hand, yields nonsense.

Categorical variables

These are made up of separate and distinct categories that do not have an inherent order. To code these variables, each category is typically assigned a value, but this assignment is *arbitrary*. Take for example the race variable. Each population group is assigned an arbitrary value. In the data set, if a person is African, the race variable for that person is set to 1. If the person is colored, the value is set to 2, Indian is 3, and white is 4. The variables have no inherent ordering or ranking.

Sex is indicated by the variable `w1_r_b4`, where males are coded as 1 and females as 2. Other examples of what we will consider categorical variables include the relationship to the head of household (`w1_r_b3`), and marital status (`w1_r_b8`).

Dummy variables

A special type of a categorical variable is a *dummy* or *indicator* variable. A dummy variable is a variable that typically takes on a value of one if the observation meets specified criteria and a value of zero otherwise. Observations that don't have the required information are assigned a missing value. In the next section, we will learn about constructing dummy variables. Later in the course you will see that dummy variables are very useful in regression analysis.

An example we gave earlier of a dummy variable is gender. In most data, people are coded as being either male or female, so a dummy variable could take a value of 1 for female and a value of 0 for male. We could either re-code the original variable (NOT recommended) or generate a new variable based on the original sex variable (strongly recommended).

In general, it is important to know the types of variables you are using because some of the tools used to analyze variables differ depending on whether the variable is continuous or categorical. Another point to keep in mind is whether the variable you are using is an individual-level or household-level variable.

Individuals and households

NIDS is a household data set. Households are sampled from all across the country from different regions in the country, while paying attention to the racial composition of neighborhoods, whether an area is rural or urban, and so on. But, households are made up of individuals: we aren't only interested in how households behave, but how people *within* households behave.

Individual-level variables

Individual-level variables are made up of values that are unique to each person in a household. An example is the variable for age (`w1_r_b6`).

To see an example of an individual-level variable, type the following:

```
sort hhid
list hhid w1_r_b6
```

As we can see, each person *within* a household has an age value that is unique to them. Other examples in the NIDS data set would include the following individual-level variables: `w1_r_b4` and `w1_r_b7` (educational attainment level).

Exercise 3

Find an individual-level *continuous* variable that you're interested in and use the `sum` command to find its mean. Report and explain it. Do not use age.

Household-level variables

Household-level variables have the same value for every person in the household. An example would be the variable for the type of toilet available to the household (`w1_h_d20`). To see an example of a household-level variable, type the following commands:

```
sort hhid
list hhid w1_r_b6 w1_h_d20
```

As we can see, all people in the same household have the same value for `w1_h_d20`.

Exercise 4

Find a household-level *categorical* variable that you're interested in and use the `tab` command to find its frequency. Report and explain it.

Generating New Variables

To use data to create new insights, we almost always have to create new variables from the variables in the data set. This could be because we do not like the way the variable was constructed originally and/or because we need to have the variable in an alternative form to analyze it. Or, we may need to create combined categorical variables or because we simply need another new variable. We shall use the following commands: `generate` , `replace`, `recode` and `egen`.

Example: Generating a dummy variable

Suppose we want to create a variable to indicate whether an individual has no formal schooling. We can use the `w1_r_b7` variable to create a new variable called `noschool` that will be equal to 1 if the individual has no formal schooling and equal to zero if the individual has some education.

To create new variables we use the `generate` command, which can be shortened to `gen`. We do this in three steps:

1. *Generate* a variable that equals 0 for *all* observations:

   ```
   gen noschool = 0
   ```

2. *Replace* the 0 with a 1 when the variable satisfies an `if` condition:

   ```
   replace noschool = 1 if w1_r_b7 == 25
   ```

3. *Replace* the 0 with a period (.) when the variable is missing:

   ```
   replace noschool = . if w1_r_b7 == .
   ```

Exercise 5

Generate a dummy variable called `male` that equals one if the individual is male and 0 if not. If the gender variable is missing for the individual the male dummy variable should be set to missing.

Once a variable has been generated we *cannot* generate it again. So we have to use the `replace` command to change the variable. Usually it will take a generate command followed by one or more replace commands to completely define the new variable. To confirm that the variable was created correctly, we can tabulate `w1_r_b7` by `noschool`, including the `missing` option to confirm that the missing values in the original variable are also coded as missing for the new variable:

```
tab w1_r_b7 noschool, missing
```

Tab and gen

Suppose we want to create a dummy variable for each race. We could create four variables, one at a time, going through the process we went through for gender and no schooling. But, there is a quicker way using the `tab` command with the gen option:

```
tab w1_best_race, gen(race)
```

This generates a dummy variable for each category of `w1_best_race`. The dummy variables are named race1 to race4, corresponding to the name we gave in the gen part of the command. To verify that `race1` is a dummy variable for Africans, type:

```
tab w1_best_race race1
```

Let's rename the race variables so we don't get confused.

```
rename race1 african
rename race2 coloured
rename race3 indian
rename race4 white
```

Recode and gen

`generate` can be paired with the `recode` command to powerful effect. Suppose we wanted to create a variable for years of completed education. We could use the generate command followed by several replace commands but a far quicker way is to use the `recode` command.

`recode` takes the existing coding of the variable, replaces it with codes that you provide, and generates a new variable names as you indicate. For education, for example, the relevant outcome that we would want to measure would be years of education, which we will call `edyears`. We will use the idea that someone will have years of education that are missing, zero, 10 years (grade 10), 11 years (grade 9), 12 years (grade 12), 13 years (some college), and so on. To do this type:

```
recode w1_r_b7 (−9/−3 24 = .) (25 = 0) (13 16 = 10) (14
    17 = 11) (15 = 12)
(18 = 13) (19 = 14) (20 = 15) (21 22 = 16) (23 = 17), gen
    (edyears)
```

Use `tab` and `sum` to investigate your new variable `edyears` and use the Stata help to make sure you know what the `recode` command does.

Transforming variables

Often it is more convenient to use the natural logs of variables because doing so *linearizes* the variable or *dampens* the effect of outliers.

It is also common in economics to use the natural logs of variables because of it allowing us to interpret coefficients in regressions as percentage changes. A typical example of this is to use something like the natural log ($\ln$) of income rather than simply income in levels. Take a look at the household incomes variable (`w1_hhincome`) using `codebook` and `sum`. How many of the observations are missing? Are there any negative values?

Now, type in the following:

```
gen lninc=ln(w1_hhincome)
list w1_hhincome lninc
```

This sequence will generate a new variable that is the log of household income.

Labeling Variables

To make your work easy to interpret for later, it is good practise to *label* your variables. This will also make it easier to generate tables for publication-ready (or assignment-ready) work later. The command to label a variable is `label` which can be abbreviated to `lab`. For example, if we wanted to label the dummy variable `noschool` that we generated earlier, we could do the following:

```
lab var noschool "D = 1 if no schooling"
```

Here the "D = 1" suggests that it is a dummy variable that is equal to 1 if the person has no schooling.

Exercise 6

Label the gender dummy that you generated earlier using the convention we have shown here.

HINT Sometimes when you copy/paste with quotation marks, they don't come out right so you may have to re-type them.

If you want to attach a label to the log of income variable we generated, you can use the label command as follows:

```
lab var lninc "Log of household income"
```

Sorting before grouping

Often we will want to see patterns in the data and arrange the data to prepare it for visualization. This often means we have to *sort* the data on particular variables, such as gender, education or age.

Suppose we want to sort the observations by age. Type:

```
list w1_r_b6
sort w1_r_b6
list w1_r_b6
```

The sort command can be used with several variables. For example, you might want to sort first on race, and then on gender and age. You would type:

```
sort w1_best_race w1_r_b4 w1_r_b6
l w1_best_race w1_r_b4 w1_r_b6
```

In the last command above, `l` is the shortest possible abbreviation for `list`.

Egen: extensions to generate

To make sense of data often requires us to make sense of the aggregate properties of subsets of the data, for example, comparing urban with rural households, or the education levels of different race groups, or the way in which household size affects consumption.

Let's say we want to create a household-level variable which contains the total number of people living in the household.

```
egen hhsize = count(pid), by(hhid)
tab hhsize
```

The egen command told Stata to `count` the number of people with non-missing values for the `pid` variable for everyone with the same value of `hhid`.

So the variable egen created is just a count of the total number of people in each household. The count is stored in the variable `hhsize`. We can see that some households have over 20 members! If we type:

```
list hhid pid w1_r_b6 w1_r_b4 hhsize
```

we see that the value of hhsize is the same within households, for each member of the household.

If we want to see a frequency table of how many households contain different total numbers of household members, we need to pick out ONE hhsize value to represent each household. To accomplish this task, the egen command is very helpful.

```
egen hh_one = tag(hhid)
sort hhid
list hhid pid hh_one
```

Exercise 7

a. Compare the output from the `tab` command using `if hh\_one == 1` to the frequency table of household size at the individual level without that option specified. What happens? Why does it happen?

b. Use your new variable `hh\_one` to calculate the mean monthly income of households in the NIDS sample. What command did you use and what is the mean income?

The tag function creates a dummy variable where the value of 1 is assigned to the first observation for each household and a 0 to any subsequent observations for that household. We can use the `hh_one` variable to generate a frequency table of household size at the household level.

```
tab hhsize if hh_one == 1
```

Complete the adjacent exercise. Counting observations is only one of the many functions egen is capable of. It can also create means, minima, maxima, medians, percentiles, to name just a few of the most commonly used functions. Here is one more example, where we use egen to expand values from one observation to other observations in the household. Suppose we have some theory saying the higher the level of education the head of household has, the more likely the other members will have high levels of education. Therefore, we want to create a variable which records for every individual in the household, the level of education of the head of the household, Try the following:

```
gen temp = edyears if w1_r_b3 == 1
egen headeduc = max(temp), by(hhid)
drop temp
```

To create a variable that contains the education of the household head, we dp the following:

1. create a temporary variable (`temp`) equal to missing for everyone except household heads, for whom it equals years of education. This means that there is only one non-missing value of temp per household.
2. use the `egen` command to calculate the maximum (`max`) of temp across all observations with the same value of `hhid`. Since there is only one non-missing value of temp in the household, which is equal to the head's years of education, the maximum of temp is simply the education of the head - but `egen` stores it in a variable called `headeduc` for *everyone* in the household. (If we had taken the minimum or mean we would have gotten the same result.)
3. we drop the `temp` variable as it is no longer useful

Type `help egen` in Stata to see more. Remember that every time you generate a new variable, its name should appear in your list of variables window.

Exercise 8

Use your `hh\_one` variable to generate a frequency table of the household head's education for each household.

Examining Subsets of Data

Sometimes, we want to work with subsets of our data. For example, we may only be interested in working adults. We use the `keep` and `drop` commands to create subsets of our data.

Before we `drop` some of our observations lets `preserve` our current data set so that we will be able to recover the dropped observations at a later stage. The `preserve` command saves our current data set in memory. We can then recover this same data set at any point by typing `restore`. Type

```
preserve
```

Suppose we are only interested in female observations. To make our data set smaller, we can

```
drop if w1_r_b4 == 1
```

When we *drop* observations, all the information associated with the observations that meet the “if” criteria will be erased from our data set, and if we want them back, the original read-only data set should be opened again (unless you have used the preserve command).

Exercise 9

What percentage of these women are under 18 years of age?

We can use the count command to see how many of these women are married.

```
count if w1_r_b8 == 1
```

Suppose we further narrow our sample of interest to adult (18 years and older) women.

Restrict the data set by typing:

```
keep if w1_r_b6 >= 18 & w1_r_b6 ~= .
```

Now using the count command, how many observations do we have left?

Now that we know how to select subsets of our data, let's go back to the original data set by typing restore. Type

```
restore
count
```

The original data set with 31,170 observations should be restored.

Scatterplots: Age and Education

We are going to compare the mean years of completed education (edyears) by race against age.

First, generate the mean years of completed education by age and race.

```
egen meaned = mean(edyears) , by(w1_r_b6 w1_best_race)
```

You can create graphs by typing commands directly in the command window or by using the menus. We start with the menus.

1. Select Graphics then select Two way graphs.
2. Click on Create
3. Under Basic plots select Line.
4. Type `meaned` for the Y variable.
5. Type `w1_r_b6` for the X variable.
6. Select Sort on X variable.
7. Select the if/in tab and in the IH: (expression) box type `w1_best_race==1`.
8. Click Accept.
9. Click Create and follow steps 3 to 6 again.
10. Select the if/in tab and in the IH: (expression) box type `w1_best_race==4`.

11. Click OK.

Stata will now generate your graph. If you look at the results screen you will notice that the following command has appeared:

```
twoway (line meaned w1_r_b6 if w1_best_race==1, sort) (
    line meaned w1_r_b6 if w1_best_race==4, sort)
```

If you had typed this command directly into the command line the same graph would have been generated.

It would be nice to have more informative labels for the graph. Let's edit the legend.

1. Select Graphics then select Two way graphs.
2. Select the Legend tab
3. Click Override default keys
4. Type the following in the box: 1 "African" 2 "White"
5. Click OK

Since there are very few observations for people over the age of 80, we could exclude those observations by adding an if statement. Click on the command line and press the page up key to make the previous command appear in the command line and edit it to look as follows:

```
twoway (line meaned w1_r_b6 if w1_best_race == 1 &
    w1_r_b6 <= 80, sort) (line meaned w1_r_b6 if
    w1_best_race == 4 & w1_r_b6 <= 80, sort)
```

Press enter and you should see a graph with the x-axis restricted to individuals younger than 80 years of age. Alternatively you can use the Graphics menus. If you select Graphics and then Two way graphs you will see that your previous specifications are still there. Click on Edit and then select the if/in tab to modify your conditions.

We could add in x and y-axis titles and other additional details to the graph. Note that we can copy the graph and paste it into Word using by right-clicking on the graph. We can also save a graph in Stata format by right-clicking on the graph and giving a file name for the graph.

When we create a graph Stata will plot a point for every observation. This may mean that there are as many as 31,170 points on your graph, many of them describing the same (x, y) coordinate. This can make the graphs slow to display on the screen and difficult to print or to copy and paste. In the case of our `meaned` variable, all observations that have the same combination of age and race will be given the same value of meaned by the egen command. We therefore only need one point on the graph to represent each age/race cell. To eliminate this problem, we can tell Stata to plot only one point per group (age and race in this case). To do this, we can use the `tag` function with our `egen` command to create a variable that equals one for only one observation *within* each age/race cell.

```
egen number = tag(w1_best_race w1_r_b6)
```

Now edit your conditions to restrict the graph to observations where `number` is equal to one.

It may be easier to get a feel for the data if we look at children of school going age and adults separately, by using if statements:

```
twoway (line meaned w1_r_b6 if w1_best_race == 1 &
    w1_r_b6 >= 6 & w1_r_b6 < 18 & number == 1, sort) (line
     meaned w1_r_b6 if w1_best_race == 4 & w1_r_b6 >= 6 &
    w1_r_b6 < 18 & number == 1, sort) twoway (line meaned
    w1_r_b6 if w1_best_race == 1 & w1_r_b6 >= 18 & w1_r_b6
     <= 80 & number == 1, sort) (line meaned w1_r_b6 if
    w1_best_race == 4 & w1_r_b6  >= 18 & w1_r_b6 <= 80 &
    number == 1, sort)
```

Saving and exporting your graph

Once you are happy with your graph you may want to save it. With the graph still open, type

```
graph save mygraph
```

where mygraph is the name you have chosen for the graph. If you want to look at a graph that you saved earlier, type

```
graph use mygraph
```

To paste the graph into a Word document you need to export it. Type:

```
graph export mygraph.pdf
```

This will save the graph as a pdf file. You can save a graph in a number of different formats. Type

```
help graph export
```

if you would like to investigate the other options.

Saving Your Data

If we've created new variables we may want to save your Stata data set in your own data folder to work with again. Click on "File, Save" in the Stata window, and save the file in your folder `/ProcessingAnalysis/IntermediateData`. Stata will warn us if we are writing over a data set with the same name. We need to be careful about whether we want to write over the existing data set. If we make a mistake you can always go back to the original data sets on Moodle.

Select solutions to coding exercises

1. Solution to re-coding the age variable where the minimum was -9, which needed to be re-coded to missing.

   ```
   replace w1_r_b6 = . if w1_r_b6 < 0
   ```

   We can now check the average to see if we have found the proper average age.

   ```
   sum w1_r_b6
   ```

2. Household income exercise. Remember that we need to specify the option `means` because that's the only information we want.

   ```
   tab w1_best_race w1_r_b4, sum(w1_hhincome) means
   ```

Commands in Lab 2

drop	keep	replace
preserve	restore	gen
sum; sum, detail	tab, sum()	egen
tag	count	max
mean	label variable	tab, gen
recode	sort	by
graph twoway	graph save	graph use
graph export	save using	–