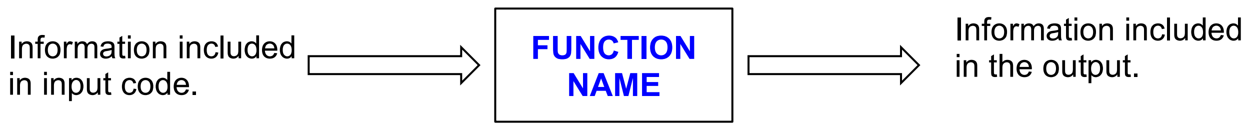
# SELF-DIRECTED STATA LEARNING

## SELF-DIRECTED STATA LEARNING

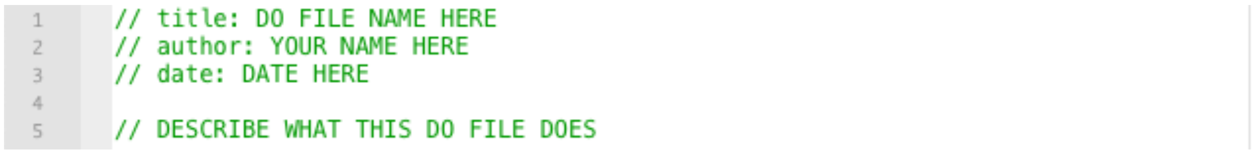
This packet will introduce you to using Stata for basic data manipulation and analysis. Like most other statistical software, Stata uses “functions” to do things. Functions are things that Stata “knows” how to do. Each function requires an input and gives you an output. When you’re coding, you’re calling on these functions to accomplish your goals with the data. We will present many different functions below. In the .do files (see explanation below), example code (see below), and this document, functions are in blue. When we introduce important functions, we will write them as a diagram with this format:


Information included in input code.
FUNCTION NAME
Information included in the output.

### Step 1. Getting started

A **.do file** is where you will enter code, execute commands, and make notes about your data analysis process. **.do files** should contain every line of the code you need to conduct a data analysis. You will save your .do file so that you can re-run your commands, edit them, use them as a template for future work etc. When you open a .do file it appears as a window in Stata.

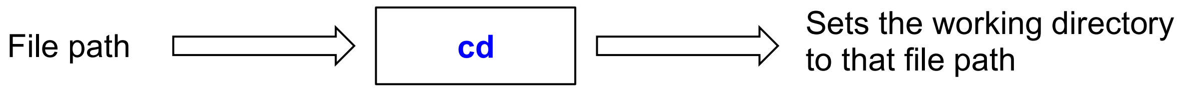
Using .do files ensures that you remember what you did during analysis and that your analysis is replicable. Like an academic paper, .do files should have a heading that includes the title of your .do file, your name, the date you last updated the .do file, and a short description of what the .do file does.

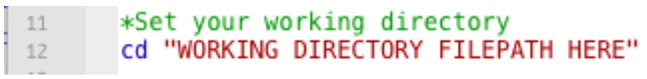


```
// title: DO FILE NAME HERE
// author: YOUR NAME HERE
// date: DATE HERE

// DESCRIBE WHAT THIS DO FILE DOES
```

Note: .do files are color coded. As we mentioned above, functions are in blue. Green text indicates notes, also called “comments,” which Stata will ignore. You can write notes on one line by starting the line with // or *. You can also write notes on multiple lines by using /* [text here] */. See example above and other examples throughout this document.

Your .do file should live in your “**working directory**.” A working directory is a folder on your computer that contains everything you need for analysis, including .do file(s), log files, and data. You should save your .do file in this folder. You will also need to set your working directory within your .do file by using the command cd (which stands for “change directory”) so that your computer knows where to find relevant files.

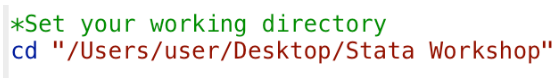



```
*Set your working directory
cd "WORKING DIRECTORY FILEPATH HERE"
```

The **working directory file path** will be unique for every person. For example, if my working directory is a folder called “Stata Workshop” that lives on my desktop, my file path would be:

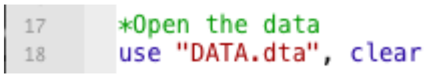
/Users/user/Desktop/Stata Workshop (for a Mac)
C:/Users/user/Desktop/Stata Workshop (for a PC)

It will look like this:

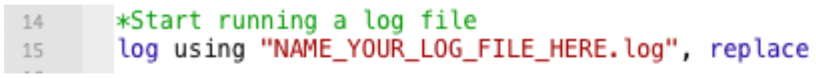

```
*Set your working directory
cd "/Users/user/Desktop/Stata Workshop"
```

(Note: To get the file path for a folder on a Mac, right-click on the folder, press Option, and click on “Copy ‘Folder’ as Pathname.” To get the file path on a PC, click on the “Home” tab and then on the icon that says “copy path.”)

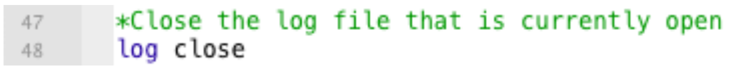
Your working directory should also contain the **data** you will be analyzing. After setting the working directory in your .do file, you should open your data using your .do file. Data files designed for Stata are saved as “.dta” files. Stata can also open other types of data files, including .csv files.


```
*Open the data
use "DATA.dta", clear
```

Finally, your working directory will also contain what’s called a **log file**. Log files keep a record of everything you do in Stata, including both the commands you type in the .do file and the output of those commands.


```
*Start running a log file
log using "NAME_YOUR_LOG_FILE_HERE.log", replace
```

At the end of your .do file, you will have to make sure to close your log file.


```
*Close the log file that is currently open
log close
```

Now your .do file is set up. The main part of the .do file will be divided into two sections, one for **data cleaning** and another for analysis. Data cleaning involves looking at and organizing the data, including creating variables. Analysis is everything else. This worksheet focuses mostly on the data cleaning section of the .do file.

Note: Once you have commands written in your .do file, you can run them in one of two ways: (1) press the “Do” button in the top right corner or (2) use keyboard shortcuts

Cmd-Shift-D (for a Mac)
Ctrl-D (for a PC)

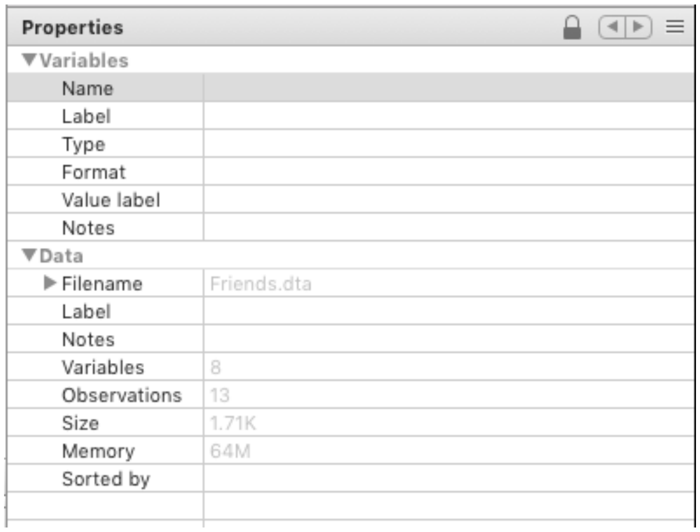
If you have certain lines of code highlighted, then these methods will only run those selected lines. If nothing is highlighted, the entire .do file will run from start to end.

**ACTIVITY 1:**

1. Download the sample .do file we created at **http://bit.ly/Do_Template**.
2. Save the .do file in a specific folder somewhere on your computer that you will use as a working directory.
3. Fill out the heading of the .do file.
4. Link to your working directory in the .do file.
5. Start a log file from the .do file.
6. Download our class dataset at **http://bit.ly/Workshop_Data**.
7. Save the dataset in your working directory.
8. Link to your dataset in the .do file and open the data.

**Step 2. Looking at your data**

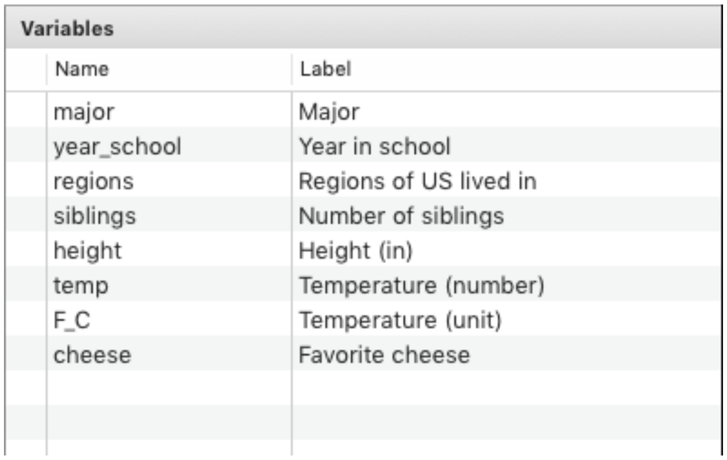
Once the data are open in Stata, we can see the number of observations contained in our data as well as the number of variables in the box called "Properties" (bottom right corner of the main Stata window). Here I am using a dataset called "Friends.dta," which is a smaller version of the dataset you will be using.



Properties

| | |
|---|---|
| ▼Variables | |
| Name | |
| Label | |
| Type | |
| Format | |
| Value label | |
| Notes | |
| ▼Data | |
| ▶ Filename | Friends.dta |
| Label | |
| Notes | |
| Variables | 8 |
| Observations | 13 |
| Size | 1.71K |
| Memory | 64M |
| Sorted by | |

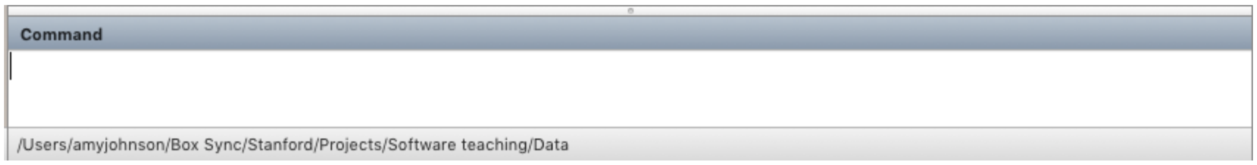
In the box called "Variables" (top right of the main Stata window), we can see all the variables in our dataset, along with the variable labels.



Variables

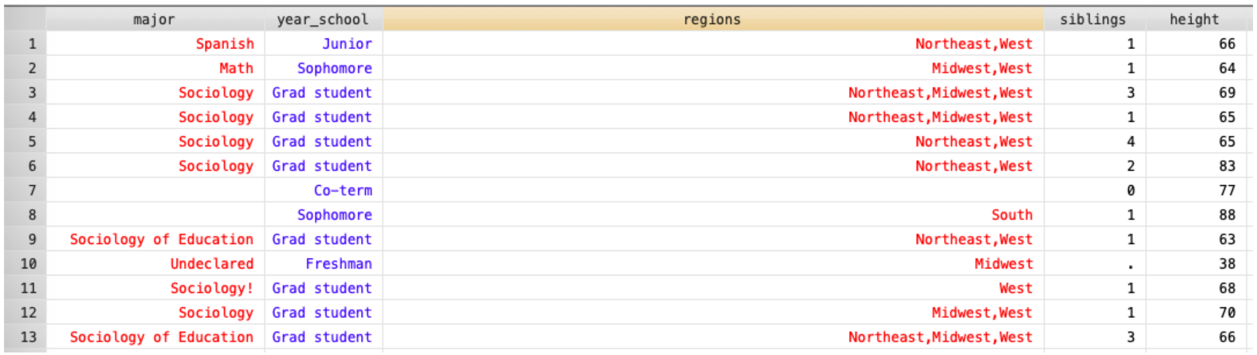
| Name | Label |
|---|---|
| major | Major |
| year_school | Year in school |
| regions | Regions of US lived in |
| siblings | Number of siblings |
| height | Height (in) |
| temp | Temperature (number) |
| F_C | Temperature (unit) |
| cheese | Favorite cheese |

At the bottom of the main Stata window is the command box. Here you can type Stata commands. We recommend using a .do file for your code, but sometimes the command box is helpful if you want to run something quickly that doesn't need to be reproducible. Directly under the command box is the file path for your working directory. Check to be sure the file path is set to the correct working directory.



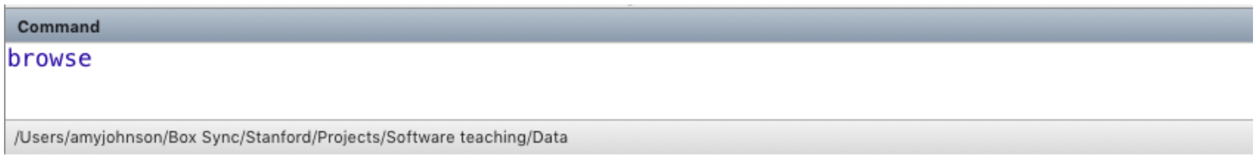
```
Command

/Users/amyjohnson/Box Sync/Stanford/Projects/Software teaching/Data
```

To look at the data, click on "Data Browser" in the toolbar at the top of the main Stata window. Each column represents a variable, with a variable name, and each row represents an observation (or, in this case, a person).



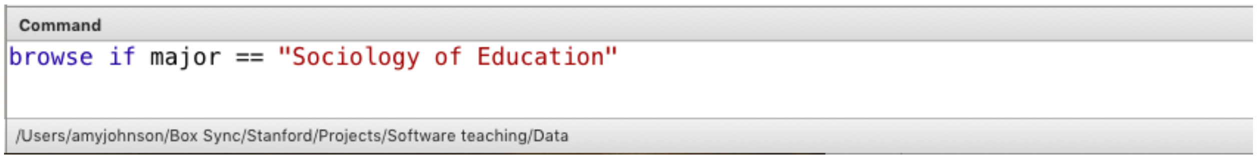
| | major | year_school | regions | siblings | height |
|---|---|---|---|---|---|
| 1 | Spanish | Junior | Northeast,West | 1 | 66 |
| 2 | Math | Sophomore | Midwest,West | 1 | 64 |
| 3 | Sociology | Grad student | Northeast,Midwest,West | 3 | 69 |
| 4 | Sociology | Grad student | Northeast,Midwest,West | 1 | 65 |
| 5 | Sociology | Grad student | Northeast,West | 4 | 65 |
| 6 | Sociology | Grad student | Northeast,West | 2 | 83 |
| 7 | | Co-term | | 0 | 77 |
| 8 | | Sophomore | South | 1 | 88 |
| 9 | Sociology of Education | Grad student | Northeast,West | 1 | 63 |
| 10 | Undeclared | Freshman | Midwest | . | 38 |
| 11 | Sociology! | Grad student | West | 1 | 68 |
| 12 | Sociology | Grad student | Midwest,West | 1 | 70 |
| 13 | Sociology of Education | Grad student | Northeast,Midwest,West | 3 | 66 |

You can also browse the data by typing **browse** in the command line and hitting Enter.



```
Command
browse

/Users/amyjohnson/Box Sync/Stanford/Projects/Software teaching/Data
```

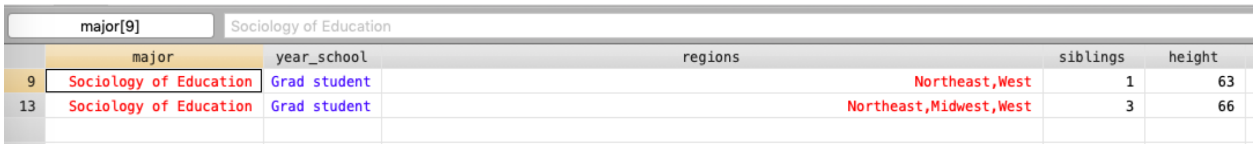
In addition to looking at the data as a whole, we can also look at specific "subsets" of the data. To do this, we can combine the **browse** command with logical if-statements. For example, if I only wanted to look at people who were Sociology of Education majors, I would type the following:



```
Command
browse if major == "Sociology of Education"

/Users/amyjohnson/Box Sync/Stanford/Projects/Software teaching/Data
```

(Note that because *major* is a string variable, I have to put the value in quotes. Also note the double equal sign, which we always use for logical statements.)

The above command will bring up a window that looks like this:



major[9] Sociology of Education

| | major | year_school | regions | siblings | height |
|---|---|---|---|---|---|
| 9 | Sociology of Education | Grad student | Northeast,West | 1 | 63 |
| 13 | Sociology of Education | Grad student | Northeast,Midwest,West | 3 | 66 |

Note that only Sociology of Education majors are included.

Stata logic syntax:

| | | | |
|---|---|---|---|
| == | "is equal to" | != | "is not equal to" |
| > | "greater than" | < | "less than" |
| >= | "greater than or equal to" | <= | "less than or equal to" |

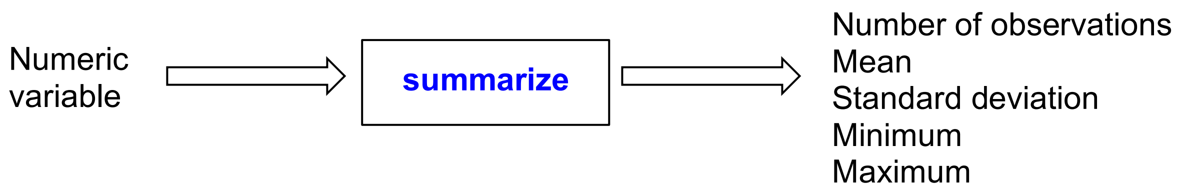
**ACTIVITY 2:**

1. How many observations are included in your dataset?
2. How many variables are there?
3. How does Stata differentiate between different types of variables? (e.g. categories, numbers, string variables)
    a. Confirm your conclusion with a partner!
4. Choose a subset of the data that's interesting to you and browse only those observations by using **browse** and a logical if-statement.

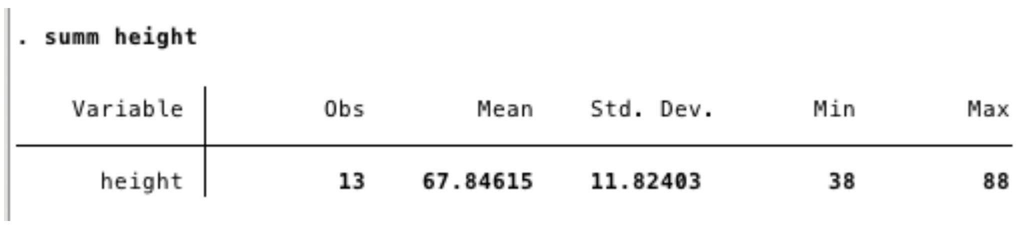
## Step 3. Summarizing variables

In social science research, we normally care about and want to analyze particular variables. The first step in analyzing a variable is to condense information about that variable across all observations. There are two main ways to condense information about a variable, depending on the type of variable.

For variables made up of numbers, such as height, we use **summarize**, or **summ**.



To summarize the variable *height* in Stata, I would type `summarize height` or `summ height` in my .do file. I could also run the command in the command box. Stata will return output like this:

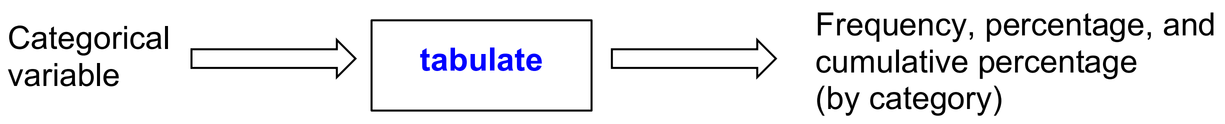


```
. summ height

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
      height |         13    67.84615    11.82403         38         88
```

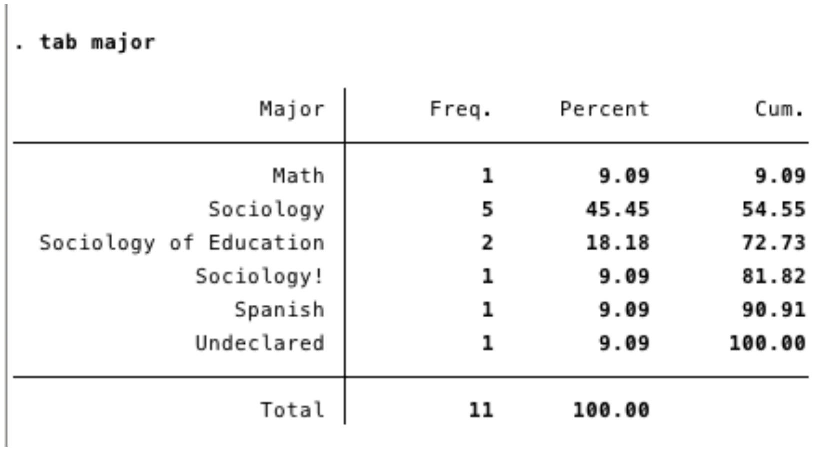
(Note that the command you typed is included above the output.)

The **summarize** command has given us the number of observations that have a value for height, and the mean, standard deviation, minimum and maximum of those values.

For variables made up of categories, such as major, we use **tabulate**, or **tab**.


Categorical variable
tabulate
Frequency, percentage, and cumulative percentage (by category)

To tabulate the variable *major* in Stata, I would type `tabulate major` or `tab major` in my .do file or command line. Stata would return output like this:

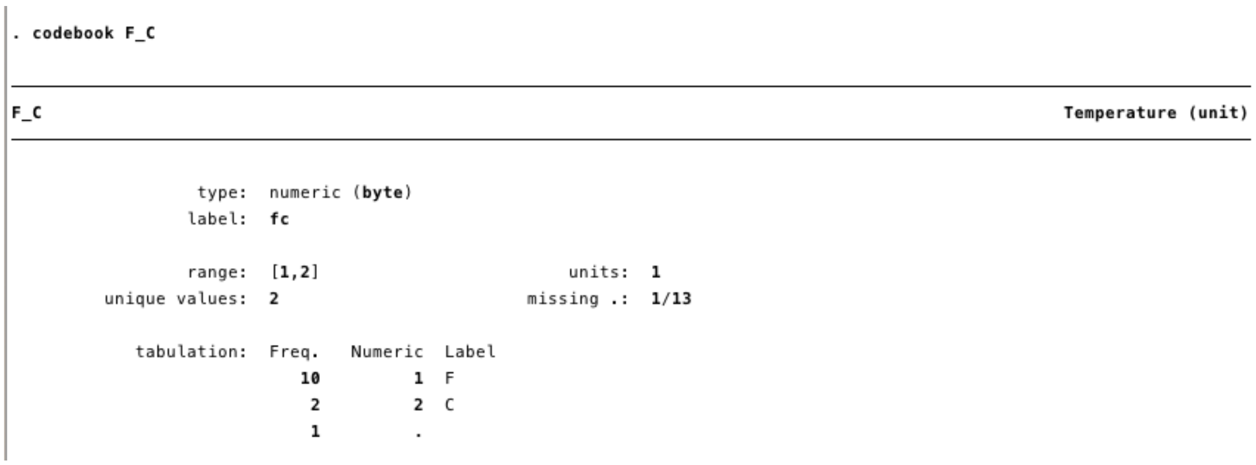


```
. tab major

                 Major |      Freq.     Percent        Cum.
-----------------------+-----------------------------------
                  Math |          1        9.09        9.09
             Sociology |          5       45.45       54.55
Sociology of Education |          2       18.18       72.73
            Sociology! |          1        9.09       81.82
               Spanish |          1        9.09       90.91
            Undeclared |          1        9.09      100.00
-----------------------+-----------------------------------
                 Total |         11      100.00
```

This tells us that 1 person majored in Math, 5 people majored in Sociology (but one person majored in "Sociology!". Note that Stata reads this as different), 2 people majored in Sociology of Education, and so on.

We can also see that 11 people in total have values of the variable *major.* But we have 13 observations overall. The mismatch between the total frequency of values for the variable *major* and the total number of observations indicates that some people are missing a value for *major*. Stata will ignore any observations that are missing information on a variable.

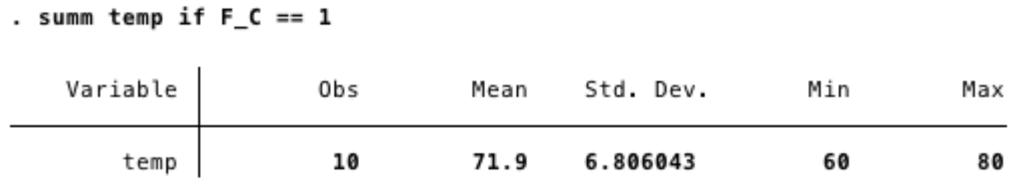
We can also subset our data, or split it into different groups. Many commands can be used on subsets of data. We subset using logical if-statements. For example, what if we only wanted to look at summary information for temperatures in Fahrenheit? This requires using two variables: *temp* and *F_C*. The variable *temp* measures the temperature value and *F_C* tells us the unit (either Fahrenheit or Celsius). Remember from browsing the data that *F_C* is a numeric variable with labels, in that the categories "F" and "C" are stored in Stata as numbers. We can see how this is done by running the command `codebook F_C` , which gives us this output:



```
. codebook F_C

-------------------------------------------------------------------------------------------------
F_C                                                                          Temperature (unit)
-------------------------------------------------------------------------------------------------

                  type:  numeric (byte)
                 label:  fc

                 range:  [1,2]                        units:  1
         unique values:  2                        missing .:  1/13

            tabulation:  Freq.   Numeric  Label
                            10         1  F
                             2         2  C
                             1         .
```

The codebook command gives us lots of information about a variable. We can see that Fahrenheit is coded as “1” and Celsius is coded as “2.” We can also see that there is one observation that is missing information on the variable *F_C* (remember that missing information is coded as “.”).

Now we’re ready to subset our data. To summarize temperatures in Fahrenheit, we would type `summ temp if F_C==1`. Note the double equal sign in the if-statement. We get the following output:



```
. summ temp if F_C == 1

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
        temp |         10        71.9    6.806043         60         80
```

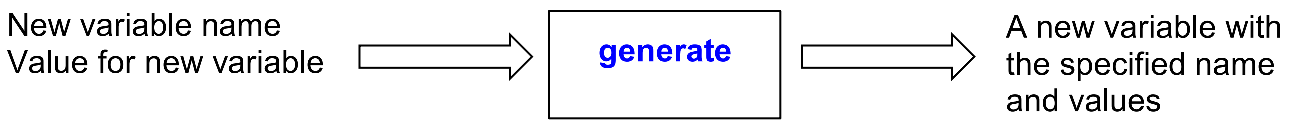
The average preferred temperature, in Fahrenheit, is 71.9 degrees.

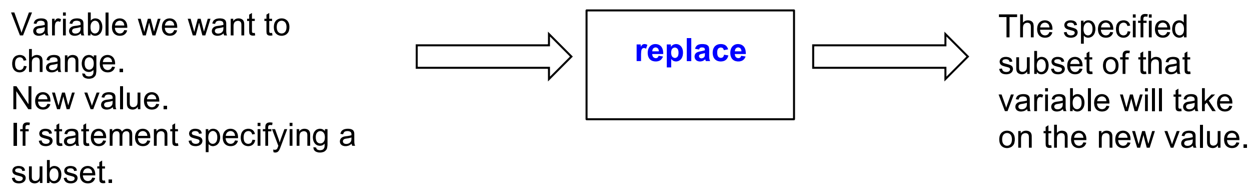
**What do I do if I get an error?**
Stata, like your 5th grade English teacher, is very particular about spelling and grammar. If you get an error when trying to run code, double check your spelling and commas!

**ACTIVITY 3:**
In your .do file...
1. Summarize the variable *siblings*.
2. Tab the variable *regions*.
3. Run the tabulate command with TWO variables listed after it. (e.g. `tab major year_school`). What happens?
4. Summarize temperatures in Celsius. What is the average favorite temperature in Celsius?
5. Look at cheese preferences for juniors.

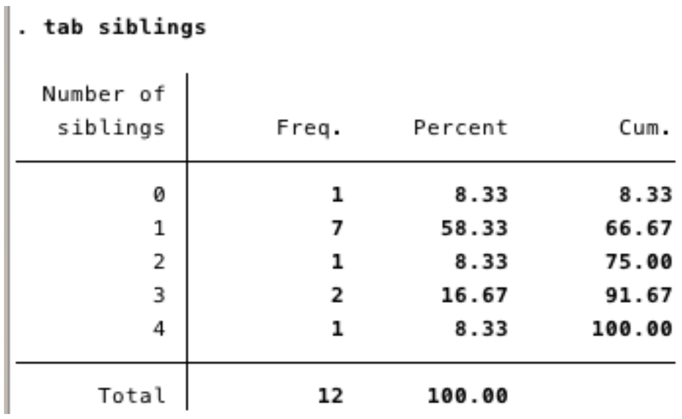
**Step 4. Manipulating your data**

Oftentimes we need to create new variables that are based on our existing variables. For example, this is helpful when a dataset has a complicated form of a variable, e.g. race, and we want to simplify it. To do this, we use two commands: **generate** and **replace**.




Variable we want to change.
New value.
If statement specifying a subset.
replace
The specified subset of that variable will take on the new value.

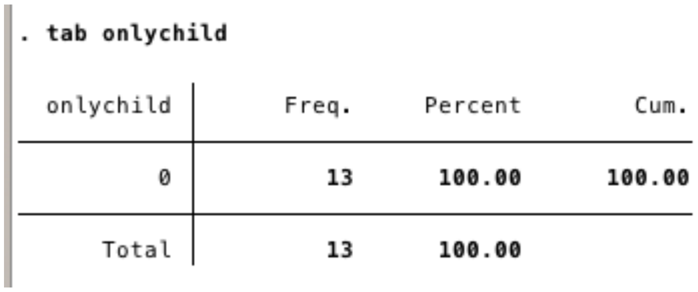
For example, say we wanted to create a variable for whether someone is an only child. We can tell who is an only child by looking at the variable *siblings*.



```
. tab siblings

  Number of
   siblings |      Freq.     Percent        Cum.
------------+-----------------------------------
          0 |          1        8.33        8.33
          1 |          7       58.33       66.67
          2 |          1        8.33       75.00
          3 |          2       16.67       91.67
          4 |          1        8.33      100.00
------------+-----------------------------------
      Total |         12      100.00
```

Only one person in my data is an only child. How many people are only children in your dataset?

To create a new variable called *onlychild*, we would first use the **generate** command. In this command, we would specify that the variable should initially be equal to 0 (indicating NO, not an only child). We would type `gen onlychild=0` in our .do file. This will assign the value of "0" for *onlychild* to every person in the dataset. To verify this, we can use **tabulate**.



```
. tab onlychild

  onlychild |      Freq.     Percent        Cum.
------------+-----------------------------------
          0 |         13      100.00      100.00
------------+-----------------------------------
      Total |         13      100.00
```

Next, we will use the replace function to re-code individuals who are only children so that they have a value of 1 for the *onlychild* variable (indicating YES, an only child). We would use the following code: `replace onlychild=1 if siblings==0`

(Note: the single equals sign assigns values to variables. Remember, we only use the double equal sign for logical if-statements.)

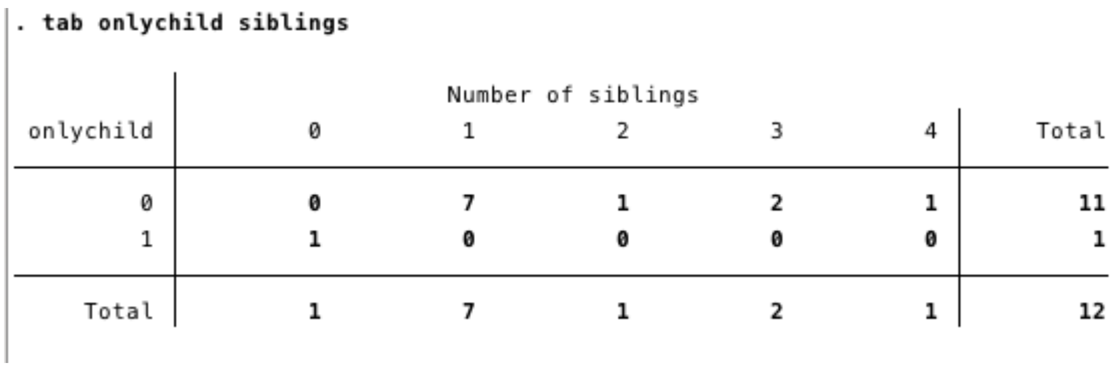
Finally, whenever we create a new variable, we need to make sure that our new variable reflects any data that were missing in our original variable(s). For example:
`replace onlychild=. if siblings==.`

In this example, we adjusted for missing data at the end of the variable construction process. Another strategy is to begin with a variable made up entirely of missing data (`generate new_var = .`) and then use replace to fill in non-missing values.

**A note about missing data**
For numeric variables (either with or without labels), missing data is coded as a period (`.`); for string variables, missing data is coded as an empty string (`""`). Sometimes, when you download data, there will be missing information that is not coded in a form Stata understands as missing (e.g., 99, “NA”, “NaN”). These values should be recoded into the appropriate form (either `.` or `""`) using `replace`.

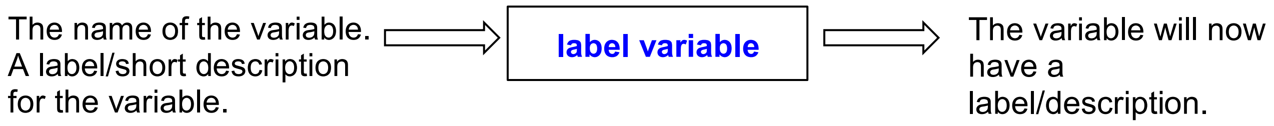
Missing data for numeric variables is also understood by Stata as “positive infinity.” This means that you must be careful when using logical operators to create new variables. For example, if you used the code `replace new_var = 1 if old_var > 2`, any observation missing on `old_var` would take on a value of 1 for `new_var`, because, in Stata’s brain, `. > 2`. You can resolve this issue by specifically excluding missing data in your `replace` function: `replace new_var = 1 if old_var > 2 & old_var !=.`

Now we have a new variable called *onlychild*. When we’re done creating a variable, it’s important to check that we created the variable successfully. For example, we should tabulate both *siblings* and *onlychild* to make sure the categories of “0 siblings” and “only child” match. In other words, we want to make sure that the same number of people are in each category. Using tabulate, we can see that one person is marked as an only child, which is the same as the person who has a 0 for the *siblings* variable.



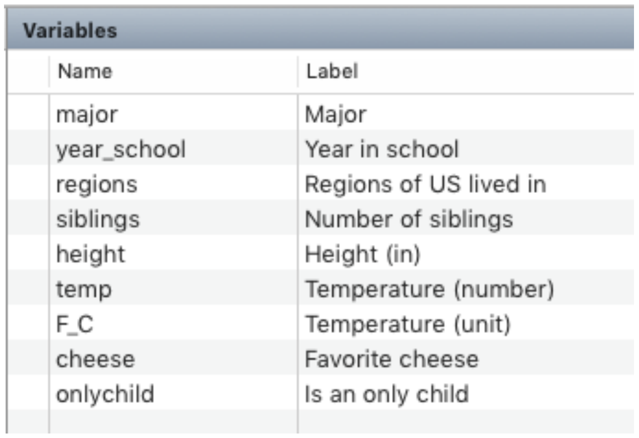
```
. tab onlychild siblings

           |                  Number of siblings
 onlychild |         0          1          2          3          4 |     Total
-----------+-------------------------------------------------------+----------
         0 |         0          7          1          2          1 |        11
         1 |         1          0          0          0          0 |         1
-----------+-------------------------------------------------------+----------
     Total |         1          7          1          2          1 |        12
```

But we’re not quite done. It’s also important to label variables so their meaning is clear. First, we have to label the overall variable using **label variable**. For example, we might label our variable *onlychild* something like “Is an only child.”


The name of the variable. A label/short description for the variable.
label variable
The variable will now have a label/description.

In Stata: `label variable onlychild “Is an only child”`

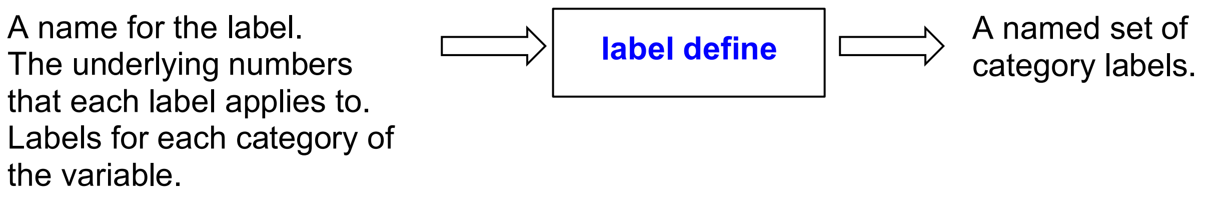
We can see that our variable *onlychild* is now labelled in the variable window:



Variables

| Name | Label |
|---|---|
| major | Major |
| year_school | Year in school |
| regions | Regions of US lived in |
| siblings | Number of siblings |
| height | Height (in) |
| temp | Temperature (number) |
| F_C | Temperature (unit) |
| cheese | Favorite cheese |
| onlychild | Is an only child |

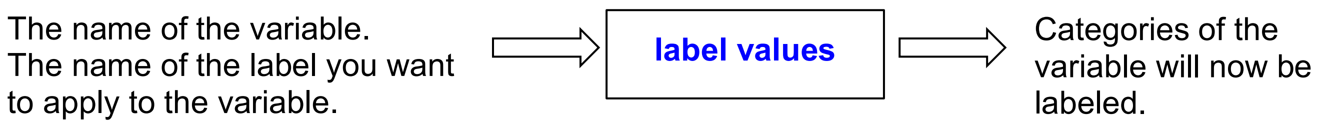
Next, if we are creating a numeric variable with labels (often called a categorical variable), we can create a label for each of the different categories within the variable using **label define**. Remember that categorical variables are understood by Stata as numbers whose values have different labels. In this case, the *onlychild* variable has values of 0 and 1, which are numbers that each represent one of the two categories. By using labels, we can clearly state what those categories are. For the *onlychild* variable, we want to label a value of 0 “Not an only child” and a value of 1 “Only child.”


A name for the label.
The underlying numbers that each label applies to.
Labels for each category of the variable.
label define
A named set of category labels.

To create a label called “onlychild_label” in Stata:

```
label define onlychild_label 0 “Not an only child” 1 “Only child”
```

This creates a category label called “onlychild_label” where the number 0 is associated with the label “Not an only child” and the number 1 is associated with the label “Only child.” Once we create the category label, we have to “apply” it to the variable using **label values**. Without applying the label, the variable *onlychild*’s values will remain unlabeled and still appear as 0’s and 1’s.

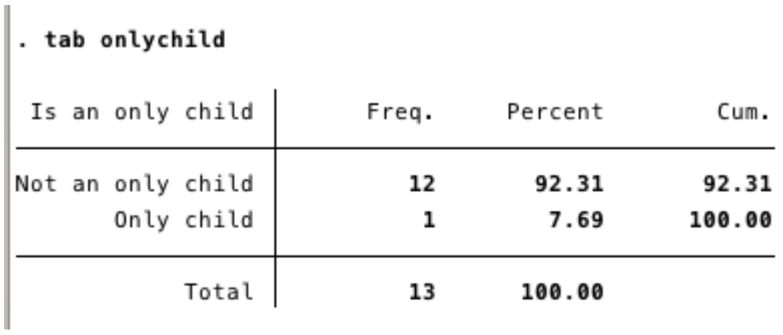

The name of the variable.
The name of the label you want to apply to the variable.
label values
Categories of the variable will now be labeled.

To apply the “onlychild_label” to the variable *onlychild*, we would type:

```
label values onlychild onlychild_label
```

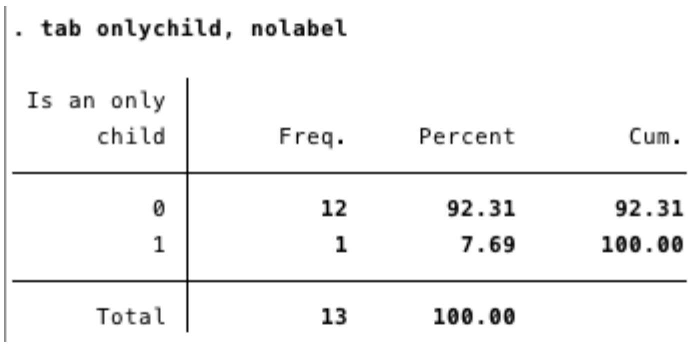
(Note that the variable we want to label comes first, followed by the name of the label we created using **label define**)

And now, when we **tabulate** the variable, we see the value labels instead of the underlying numbers:



```
. tab onlychild

  Is an only child |      Freq.     Percent        Cum.
-------------------+-----------------------------------
Not an only child  |         12       92.31       92.31
       Only child  |          1        7.69      100.00
-------------------+-----------------------------------
            Total  |         13      100.00
```

If we want to see the underlying numbers after applying our value labels, we can **tabulate** the variable and specify the option “nolabel”:



```
. tab onlychild, nolabel

 Is an only |
      child |      Freq.     Percent        Cum.
------------+-----------------------------------
          0 |         12       92.31       92.31
          1 |          1        7.69      100.00
------------+-----------------------------------
      Total |         13      100.00
```

**A note about options and help pages**
Many Stata functions have **options**, or ways to edit the function's output. Options are included after a comma following the main function. You can see different options, as well as more information about any function, using the **help** command. Type "help" and the name of any Stata function (e.g. **tabulate**). What happens?

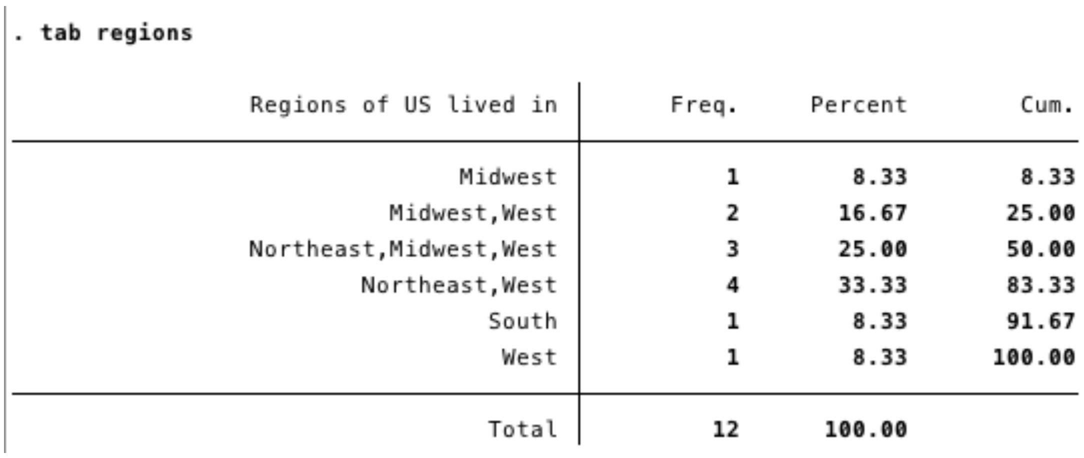
**ACTIVITY 4:**
Note: this is challenging! Work together.

1. Create a variable for type of cheese (soft or hard) using if statements and generate/replace. Be sure to take missing data into account, if necessary.
    a. Label the variable
    b. "Tabulate" this variable. What does it look like?
    c. Create value labels for the categories of the variable and apply them to the variable. Now tabulate the variable. What does it look like now?
2. Look up the help page for the function **summarize**. You should see an option called "detail." What happens when you summarize the variable *height* specifying the "detail" option?

## Step 5. Other helpful functions (OPTIONAL)

So far, we have walked you through some functions necessary for data cleaning. But Stata has LOTS more functions. Below are two examples. Try them out on your own. As you learn new functions, you will learn what each function does and the information it requires.

*Example 1*. **strpos** for string variables
String variables can be challenging. For example, look at the variable *regions*.

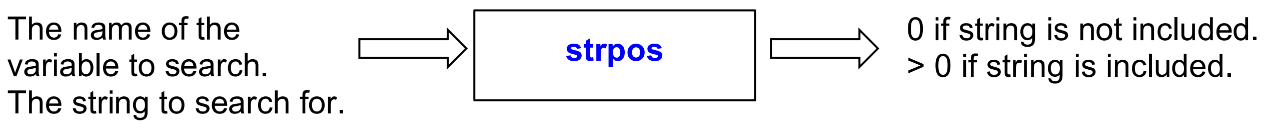


```
. tab regions

    Regions of US lived in |      Freq.     Percent        Cum.
---------------------------+-----------------------------------
                   Midwest |          1        8.33        8.33
              Midwest,West |          2       16.67       25.00
    Northeast,Midwest,West |          3       25.00       50.00
            Northeast,West |          4       33.33       83.33
                     South |          1        8.33       91.67
                      West |          1        8.33      100.00
---------------------------+-----------------------------------
                     Total |         12      100.00
```

Each category lists multiple regions. But what if we want to look only at people who have ever lived in the Northeast? How would we subset those individuals? One way would be to use OR statements to list every possible category of regions that includes "Northeast":

```
browse if regions=="Northeast,Midwest,West" | regions=="Northeast,West"
```

But this would be very time-consuming, and much harder when we have more data. Another option is to tell Stata to look within a string variable. We can use the function **strpos** to do this.


The name of the variable to search.
The string to search for.
strpos
0 if string is not included.
> 0 if string is included.

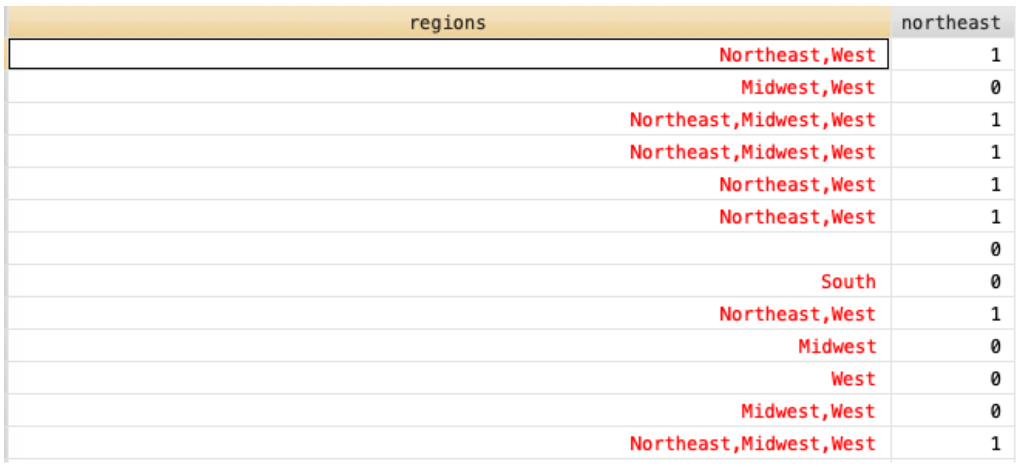
This function can be tricky, so here is an example. Say we want to create a binary (0/1) variable called *northeast* that tells us whether someone has lived in the Northeast.

First we would use `gen northeast=0` to create the variable.

Then we would use the following code:

```
replace northeast=1 if strpos(regions,"Northeast") > 0
```

Now if I type `browse regions northeast` I will bring up the data browser with the original variable *regions* and also the new variable I've created, *northeast.* What do you notice?



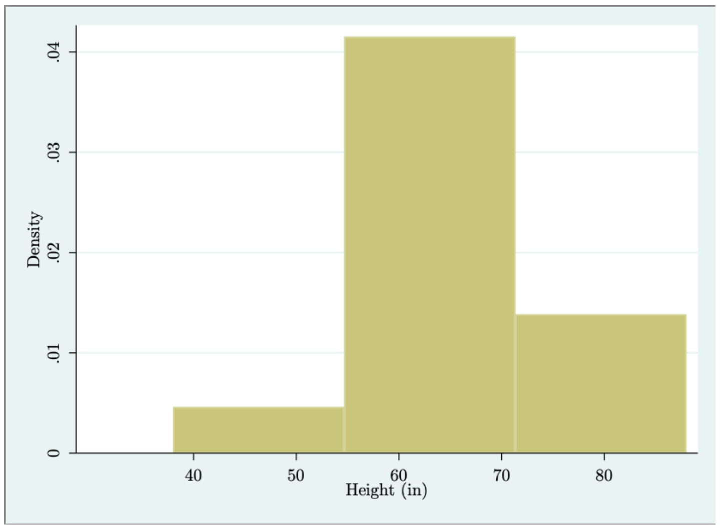
| regions | northeast |
|---|---|
| Northeast,West | 1 |
| Midwest,West | 0 |
| Northeast,Midwest,West | 1 |
| Northeast,Midwest,West | 1 |
| Northeast,West | 1 |
| Northeast,West | 1 |
|  | 0 |
| South | 0 |
| Northeast,West | 1 |
| Midwest | 0 |
| West | 0 |
| Midwest,West | 0 |
| Northeast,Midwest,West | 1 |

*Example 2.* Graphing with **histogram** and **scatter**
A key part of data analysis is creating **data visualizations**, or graphs. Data visualizations can show information about one variable or the relationship between two variables. Two common forms of data visualization are **histograms** and **scatter plots**.

Histograms show the distribution of a numeric variable. We create them using the command **histogram**.

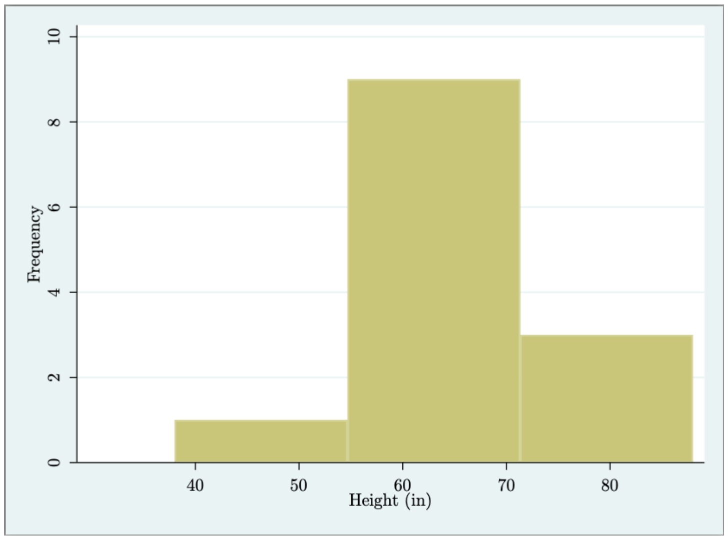
The name of the variable to plot. Whether you want to show the density or frequency. $\Rightarrow$ $\boxed{\textbf{histogram}}$ $\Rightarrow$ A histogram

Let's say we wanted to show visually how tall people in our sample are. We could create a histogram for the variable *height*. We can use the command
`histogram height, density` and Stata would output:



(Note that there is a comma before we type "density" because "density" is an **option**.)

Alternatively, we can use the "frequency" option instead. We type the command `histogram height, frequency` and Stata would output:

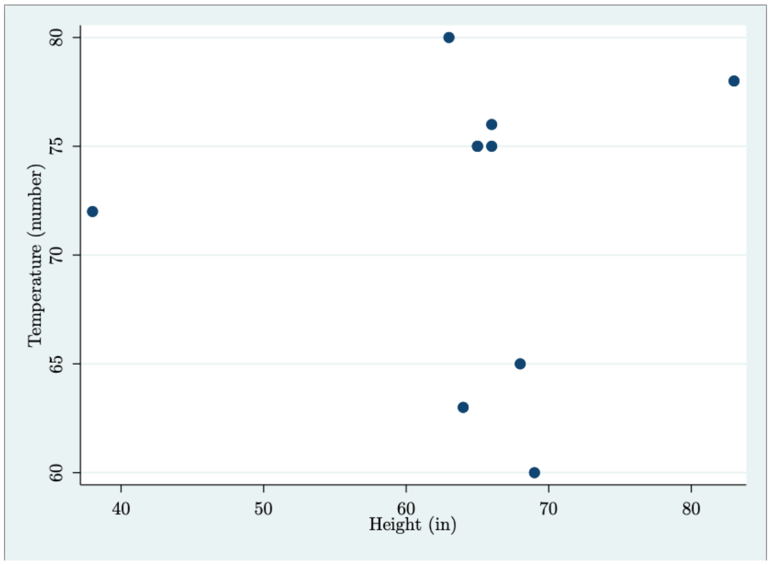


The difference between the two is what Stata puts on the y-axis. When we specify density, the y-axis is a measure of “density” for each range of *height* values. Don’t worry right now what density means (no one really knows). When we specify frequency, the y-axis is a count of how many people fall into the range of *height* values. This is more intuitive and useful.

While histograms are useful for information about one variable, sometimes we want to show information about two variables. Scatter plots show the distribution of two continuous variables. To create one we can use the command **scatter**.

$$\begin{tikzcd} \begin{array}{l}\text{A variable for the y-axis}\\ \text{A variable for the x-axis}\end{array} \arrow[r, Rightarrow] & \boxed{\textbf{scatter}} \arrow[r, Rightarrow] & \text{A scatter plot} \end{tikzcd}$$

Let’s say we wanted to show the relationship between someone’s height and their ideal temperature (in Fahrenheit). We would use the command
`scatter temp height if F_C==1` and Stata would return the output:



Because we typed *temp* first, it is on the y-axis. What would happen if we typed `scatter height temp if F_C==1`?

(Note: Why do we limit this scatter plot only to the subset of observations for which F_C==1?)

**ACTIVITY 5:**

1. Create and label a variable called *midwest* for whether a person has ever lived in the Midwest using **strpos** and **generate**/**replace**.
    i. Summarize the variable *siblings* for people who have ever lived in the Midwest.
2. Create and label a variable for whether a person has lived in both the Midwest and the West. Summarize the variable.
3. Create a histogram for favorite temperature in Fahrenheit.
4. Create a scatterplot showing the relationship between someone's height and the number of siblings they have.