Training 1

This training is the first in a series of trainings to help you learn how to use STATA. Each training is designed with specific objectives. They are designed for you to read through the training document while you are doing the work on your own in STATA.

After each training, there will be an application exercise for you to practice what you have learned, often combining what was done in previous trainings. You should complete the application exercise using a do file in STATA. You can then write your answers on the application exercise document. These two documents, the do file and using the do file to write out answers and interpretations are important to use in all your research.

Here are the objectives for Training 1:

Objective 1: Introduce to the STATA layout

Objective 2: Loading different files into STATA

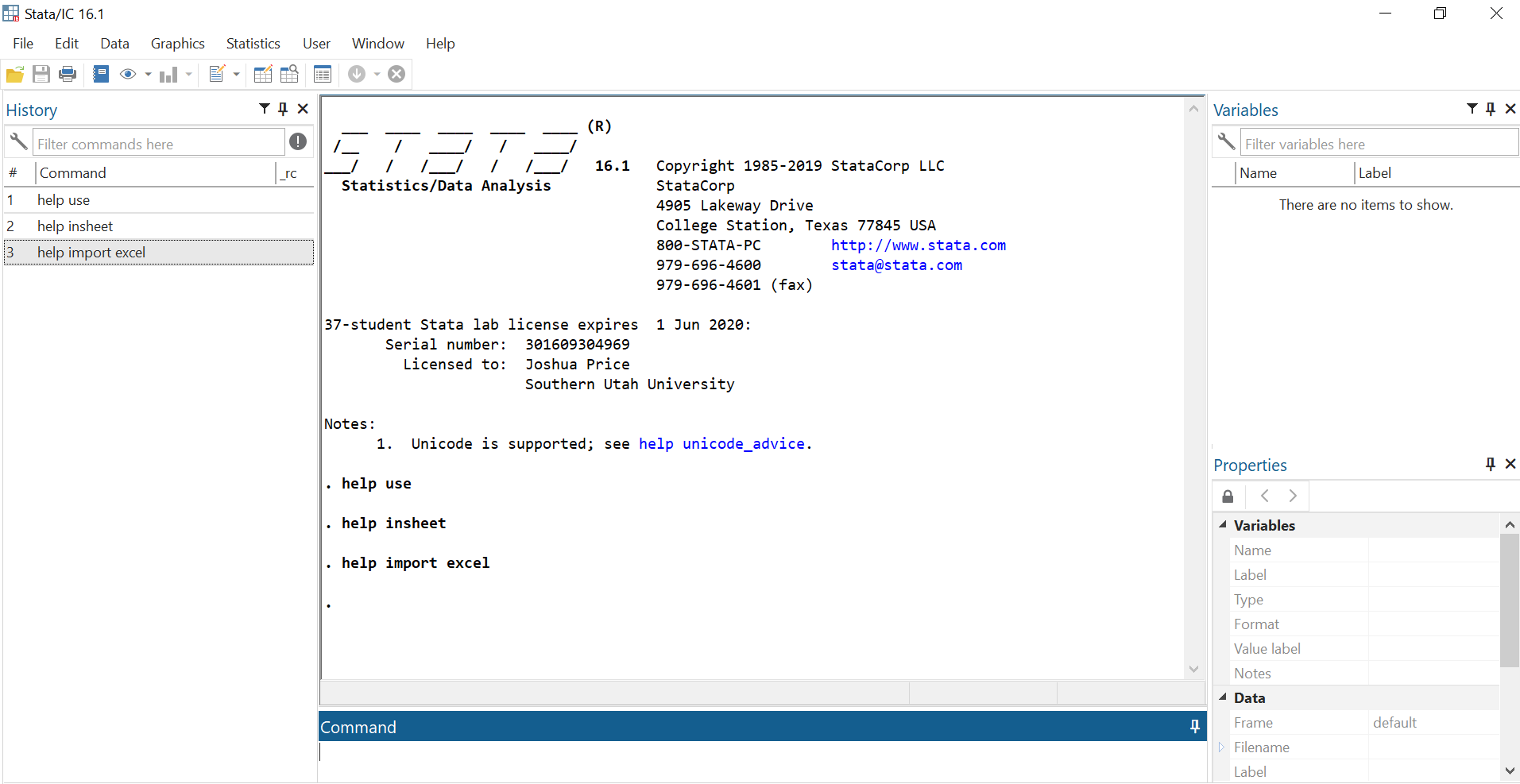
Objective 3: Estimate Summary Statistics

Objective 4: Inference Testing

Objective 1: STATA Layout

Browse/Edit – You can click on this to view the using data in a spreadsheet

Do-file Editor – Click on this to open a do file. You will do this all the time, become friends



History – will show you what previous commands

Variables – This will list the variables that are in the using data set

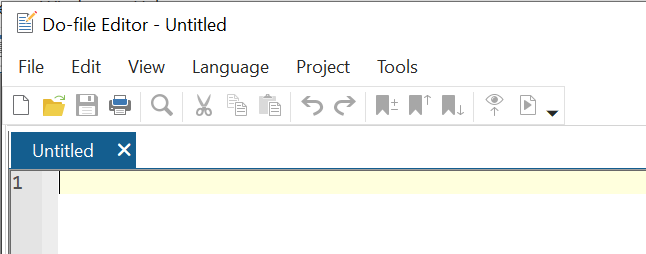
This is the display that will show the results of the commands that you have run

Command – Here is where you can insert commands one at a time

**Do-file Editor**

If you click on the “New Do-file Editor” icon in the ribbon, it will open a new window that will serve as your do-file. A do-file is where you can write your code to conduct your analysis. Using do-files is an important part of research for several reasons:

1. It allows you to keep a record of all your code in a single file
2. It allows you to identify errors and correct for them without having to re-do your previous commands
3. It allows you to save your progress and come back to it later
4. It allows you to save your do-file with others so they can see how you conducted your analysis.



Here is where you type in your code. Line by line, it’s gonna get real

When you want to run your code, click the run button. It will show up in the STATA display in the other window. Or you can use a hot key “Ctrl + d”

We don’t have anything to type in yet, but you want to always get in the habit of using a well annotated do file

**Command Prompt**

There will be many times where you want to quickly type in commands without having to run your entire do file. That can be done by going back to the main STATA window and typing in commands into the command prompt. You can type one command at a time and each time you hit enter, it will execute your command.

**History**

If you use the command prompt, it will save each command in the Command History on the left. If you single click on a previous command, it will put that into your command prompt. Do this if you want to modify or edit a previous command. If you double click the previous command, STATA will execute the command. If the command in the history is red, that means there was an error in executing that command.

Objective 2: Loading Files

STATA is a software program used to analyze data. In order to do this, you need to load data into STATA. STATA can upload many different types of files, but there are three that are most common. We will walk through how to load each of these one by one, but first, let’s learn our first command: change directory.

**Change Directory**

The change directory command tells STATA the folder from where you want to import files into STATA and the folder where you want STATA to export files. Here is an example of a change of directory command:

cd G:/Price

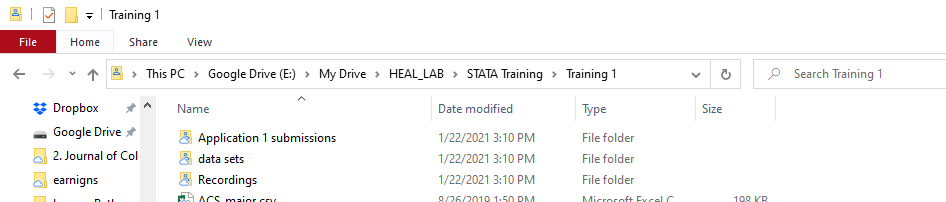
This will change the directory to the G drive and find the folder Price. As we import data sets, you want to change the directory to where those data sets are located. Let’s suppose that the data sets are located in a subfolder called “data sets”. Here is an example of the command:

cd G:/Price/data sets 🡨 BAD!!!!

If you run this command, you will get an error r(198), invalid syntax. That is because there is a space in the folder name. So you can either not put spaces in your folder and file names, or you can use quotation marks to indicate the entire folder or file

cd “G:/Price/data sets”

Where there are a lot of subfolders, I have found that I go to file explorer and copy and past the folder location. If you click on the folder location, it will highlight the path to the folder. Just copy and past that directly into your do file.



**Import Files**

The following summarizes the three common file types and the commands needed to import them into STATA.

|  |  |  |
| --- | --- | --- |
| File type | Description | Syntax |
| .dta | This is a STATA formatted data set. | use *filename.dta*, clear |
| .csv | Comma Separated File | insheet using *filename.csv*, clear |
| .xlsx | Excel file | import excel using *filename.xlsx*, first sh(“name”) clear |

The syntax is the code that you have to type in order for STATA to load the data set. Generally speaking, syntax requires three parts:

1. the command
2. details on what the command will do
3. options

Looking at the syntax for how to load a .dta file

1. The command is use
2. the details require you to identify the name of the file. If you have changed your director to where the file is located, a file name is sufficient. If you want to use a data set that is in a different folder than your director, you do not need to change your director to access it. you can type in the path and file name. For example:

use “G:\Price\data\attend.dta”, clear

1. the option here comes after the comma, clear takes whatever data set that was previously loaded in STATA and clears it so you can load a new data set.

Notes:

* When you upload a new data set, you need to clear out the old data set. That is what ,clear does.
* When using .dta files, you do not need to include the .dta, you can just use the file name (see example on next page)
* For import excel, the options include do the following:
  + first – treats the firstrow of the excel spreadsheet as variables
  + sh(“name”) – the excel spreadsheet might have multiple sheets, you can identify the sheet you want to import. If you don’t use this option, the default is the first sheet
  + Here is an example of how to import an excel file



* The order of the options is not important

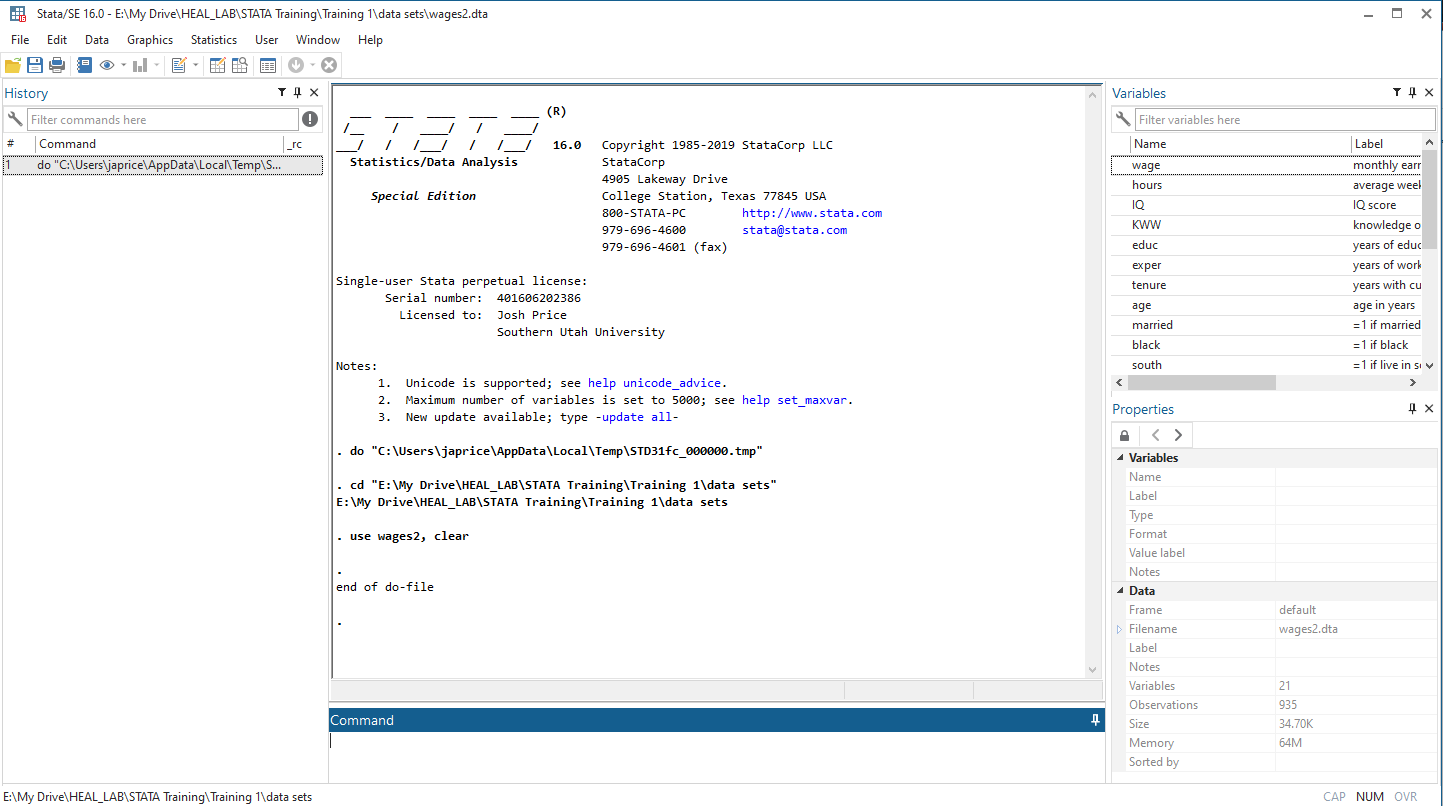
For the rest of this training, we will use a data set called wages2.dta. Here is the syntax to load that data set.

cd "E:\My Drive\HEAL\_LAB\STATA Training\Training 1\data sets"

use wages2, clear

If you do this in a do file, you will see several things on the main STATA window:

1. you will see the output of the line by line code
2. you will see the variables in the data set show up on the right hand side
3. you will so the do command show up in the command history



What you will not see is the underlying data in a spreadsheet format. You can look at it by clicking on the Data-Editor (Browse) icon in the ribbon. There is also the Data-Editor (Edit) icon right next to it. I discourage you from clicking on this link. It will show you the same data, but now you can make changes to the data and they will be saved. Any changes you make to the data should be documented through a do-file. Browse will let you see the data but you won’t be able to edit it.

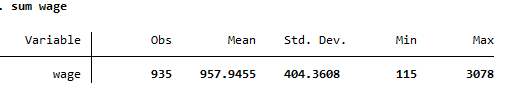
Objective 3: Summarizing Data

When we say summarizing the data what we mean to provide numbers or figures that explain parts of the data. For example, a mean shows the expected value of a variable. A standard deviation shows the spread or variance of the data. To estimate these two numbers, we can use the command summarize (Note: you should have loaded wages2.dta into STATA). It goes like this:

sum *varlist*

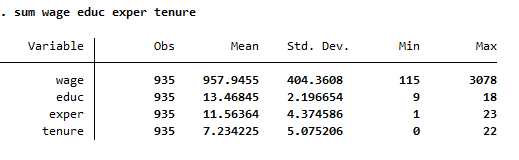
*varlist* refers to the list of variables you want to summarize.

If we want to summarize the variable for monthly earnings, we can do that in the following way:

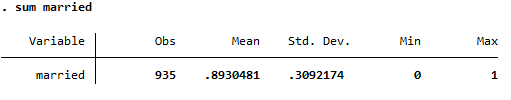


In addition to mean and standard deviation, it also provides the number of observations and min and maximum values of the variable.

You could even summarize several variables in a single line of code:



You can even summarize a binary variable, one that takes on the values of 0 or 1. For example, married equals 1 if the respondent is married and 0 if single.



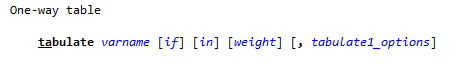
This means that 89.3 percent of the sample is married.

Rather than the mean and standard deviation, you might want to see the distribution of the variable. That is, you want to see how many observations there are for each potential outcome of the variable. This can be done with the tabulate command.

Note: If you want to learn more about a command, you can type help then the command you want to learn more about. I advise that you do this right in the command prompt.

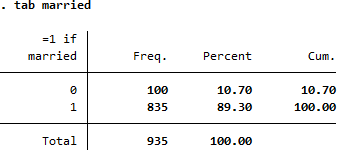
This will open the codebook that will show you the syntax, the options, and incredibly important, near the end of the file you will see different examples of how to use the command.

If you got to the codebook for tabulate you will see that there are several types of tabulate. The one we are doing here is one way. You will see the following in syntax:



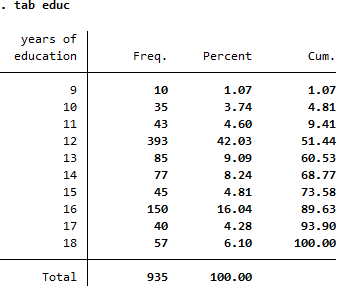
The underline in the command shows the fewest number of characters you have to type to run the command. While we could just type, ta, I tend to type, tab, as a short-cut for the tabulate command.

Here is an example of using tabulate on a binary variable:



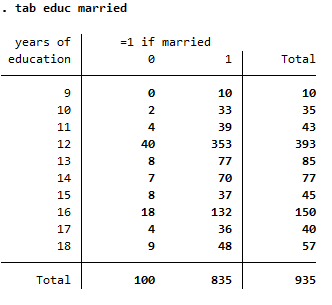
Tabulate breaks down the variable by the number of observations per outcome. It provides both the probability density probability and the cumulative density probability.

It can also work with a variable with several outcomes, like education.



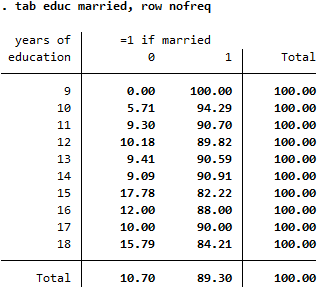
This shows the entire distribution of the variable. This can also be shown with a histogram; this will be talked about in Training 6: Creating Graphs.

You can also Cross-tabulate two variables to show a distribution between the two (This is an example of tabulate two-way from the codebook). For example:



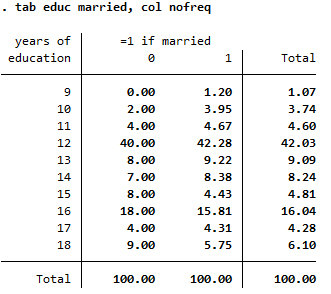
Reading through the codebook can give you different options and ideas of how to use the command. I highly recommend that you become comfortable reading through the codebook. While you may use a command a certain way, reading through the codebook can help you understand different and even better ways to use commands.

Here are some options you can use with the tabulate command.



This shows the percentages by row and does not display the frequencies or the number of observations.

Look at those with 12 years of education, among those individuals, 10.18% are single and 89.89% are married



This shows the percentages by column and does not display the frequencies or the number of observations.

Among those that are single, 40% have exactly 12 years of education. Among those that are married, 42.28% have exactly 12 years of education.

Objective 4: Inference Testing

In this section we will conduct two types of tests, testing against a single population parameter and testing two sample parameters. This is not a summary of the theory of inference testing, that is up to you to know. Here is a link to brief description of inference testing ([link](https://www.youtube.com/watch?v=kAVZVg4rUdY&list=PLHhgD2Ce826DK3i7RqY3UNcW9kDFmCN6K&index=6)). This is one of the most important principles in data and regression analysis, so make sure you have a solid understanding of inference testing, hypothesis testing, t-stats, p-values, and reject/fail to reject interpretations.

**Test against a single population parameter.**

Consider someone who claims that the average years of education of the population in the United States is 13 years. If you have a data set that is a nationally representative sample, you can test this claim. Let’s assume that wages2.dta is a nationally representative data set.

Start with a null hypothesis:

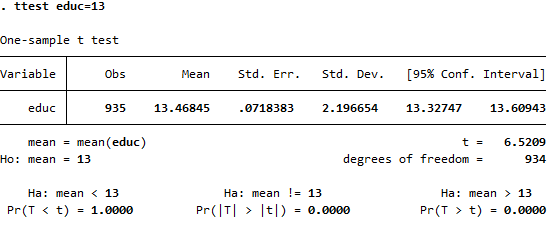
In this example, is a sample parameter (the sample mean, ) and 13 is a population parameter (the population mean, )

We can test this by forming a t-test:

We can do this in STATA, by doing a t-test

ttest education = 13

STATA will return the following output



From this we can focus on two things:

**t-stat**

we compare the t-stat to a critical value. If the t-stat is above the critical value then you would reject the null

Here are three critical values that are used often:

10% level (alpha = .10) critical value = 1.645

5% level (alpha = .05) critical value = 1.96

1% level (alpha = .01) critical value = 2.576

5% level is the most commonly one used, but later we will use all three.

In this case we compare t-stat = 6.52 > 1.96 critical value 5%. So, we reject the null. The null is that the average years of education is 13. We reject that and provide evidence to the contrary. So that we can say that the average years of education is not 13 years.

**p-value**

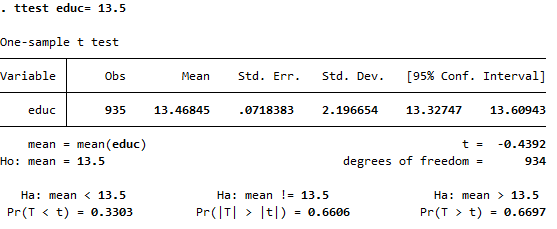
we compare the p-value to the alpha level that we choose. If the p-value is less than the alpha level, then you reject the null. Here the p-value is 0.000 which is less than an alpha of .05. So we reject the null. The null is that the average years of education is 13. We reject that and provide evidence to the contrary. So that we can say that the average years of education is not 13 years.

Here’s another application of testing the sample parameter against a population parameter. Suppose that the census shows that the average years of education is 13.5 years. The census is a survey of the entire population.

Now suppose you have a data set and you are not sure if it is representative of the national population. You can use the census to know the population parameter and then test whether your sample is the same.

Start with the null hypothesis again:

We can test that in STATA:



**t-stat**

here the t-stat is less than the critical value of 1.96 (alpha = .05). Note: it is less than the absolute value.

**p-value**

here the p-value of 0.6606 is NOT less than .05.

Either way you look at it, the conclusion is the same. You fail to reject the null. So, we conclude that our sample is no different than the population. This is good news because it provides evidence that our sample is not different from the population.

**Test two samples parameters**

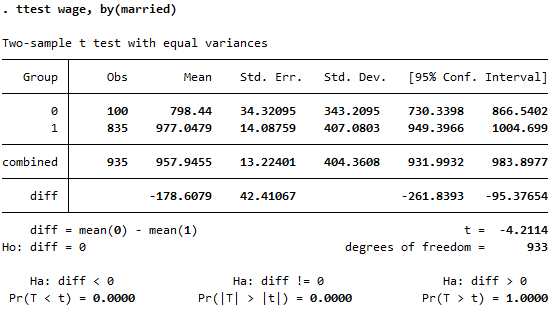
This test is used to test if two parameters in the data set are equal to each other. Questions that motivate this type of test might include: Are there more care accidents on rainy days than sunny days? Are there gender differences in high school GPA? Do student who have a job in college have worse grades than those that don’t?

The basis of these questions is to test the mean of a continuous variable based on a binary variable. Number of car accidents (continuous variable) based on a binary variable, rainy day equals 1 and non-rainy day equals zero.

Here is another example that we can test with our data set: do single and married individuals have the same wages?

We can set up a null hypothesis in the following way:

We can test this in STATA:



We can do the same comparisons, t-stat to critical value and p-value to alpha.

**t-stat**

t-stat is equal to -4.2114 which is less than -1.96 (critical value of a two-tailed test with alpha of 0.05. We would reject the null hypothesis

**p-value**

We look at the two-tailed test p-value, that’s the middle one, and see that it is 0.000. This is less than alpha (0.05), which leads us to reject the null.

Either way, we reject the null.

This means that wages are NOT the same between single and married individuals. Or we can say that married individuals make more money than single individuals.

You should be able to conduct your inference testing without a pre-conceived alpha. Rather than always use alpha=0.05, be comfortable with using the different values of alpha or confidence intervals to conduct your testing.

You have now completed Training 1. Congratulations! Hopefully you followed along and have created your own do-file as you achieved the four objectives in Training 1. It is incredibly helpful to save these do-files in a way that you know where to find them. Many times, you will be working on a problem and realize that you wrote a do-file that would help you solve this exact problem. You can then go back to that do-file and see the code you wrote.

Conclusion

This is only possible if you are organized in how you save your files. Find an system that works for you. For me, it consists of the following for these trainings.

* Folder called “HEAL\_STATA\_TRAININGS”
* Subfolder for each training
  + Training\_1
  + Training\_2
  + Training\_3
  + Training\_4
* Within each Training folder, I would have 3 additional folders
  + data
  + do\_file
  + output

**Data Folder**

I would download all data files into my data folder. I will always keep a copy of the original data set. This is important, do not makes changes to your data set in STATA and then overwrite the original data set. If you make changes, save the new data set with a different name.

**Do-file Folder**

I would save my do-file (at times there may be more than one do-file) in the do-file folder. When working on research projects, I find that over the course I modify, change, add to, and even subtract from the code in my do files. As a result, as I work on my do files and makes changes, I always save as under a new file name. If you were to open one of my do-file folders you might see the following:

* create\_master\_data\_set\_v1.do
* create\_master\_data\_set\_v2.do
* create\_master\_data\_set\_v3.do
* create\_master\_data\_set\_v4.do
* analysis\_v1.do
* analysis\_v2.do
* analysis\_v3.do
* analysis\_v4.do

**Output Folder**

While we didn’t use it in this training, I create a folder that will hold all the files that I output from STATA. If I output an excel file from STATA (like outreg2 after a regression) I put it in this folder. If I make changes in Excel, then I save that file in a different folder. I have a folder that I call Tables, where I save any excel file that I have modified by hand.

I hope that this type of organization will help you as you become a STATA Jedi. The first task for you as a young STATA padawan is not to complete Application 1.

Best of success.