Stata handout 3

# Do-files

When you work in Stata, you only rarely use the command window that we have been using in our classes. Instead, you write scripts for Stata that contain multiple lines, and save them into **.do files**. You can think of do-files as small snippets of codes (I would not call them ‘programs’ in the modern sense). You can write and edit do-files in the **do-file editor** in Stata. (Look for the editor’s icon in your open Stata window.)

The first thing you do with a do-file is clear the memory of Stata with the ‘**clear**’ command.

Then you possibly change the directory using the ‘**cd**’ command, just like we practiced. This is a best practice, even though some lazy people may leave this step out.

Then you usually open a log by typing ‘**log using** *chosen\_filename*, t replace’. This should be also familiar, just we have now an additional option (besides the ‘t’, which makes Stata generate a portable .txt file as log): replace. This is usually there for the do-files, as you may end up running the code many times, and of you do not put the ‘replace’ option for Stata, it will not overwrite the existing log-file you created from the previous run.

You should also write ‘**log close**’ right away at the bottom of your file, as you should not keep a log open between runs. This also means that if you have an error in your code, and Stata stops the execution of the do-file in the middle of the file, you need to type ‘log close’ to the command window to close the log you opened in the log file manually before another re-run. (Stata is just going to give you an error message if you try to open a new log before closing the previous one.)

Do-files are important, because they make our work transparent, as you can always rerun them with no additional cost.

# Comments

**Comments** are text in a do-file (or in the command line for that matter) that are NOT executed by Stata. These are messages to the reader of your code (e.g. your future self!). It is very important that your code is well-commented in order to be able to understand it after a couple of months you wrote it. You can see something is a comment, as the .do-file editor will use a differently colored font (typically light green).

To ‘comment out’ (the rest of) *a line* (tell Stata: “this is not a code, do not try to execute it; just put it in the output as is”), you can either type ‘//’ or ‘\*’ (the asterisk is from Shift+8). Whatever you write after these key-strokes is going to be a comment, until you hit ‘Enter’ to start a new line.

If you would like to write multiple lines of comment, you can type ‘/\*’ that opens a comment, write your comment, and close it by typing ‘\*/’. As a not particularly great practice, some just keep starting every comment line by an asterisk.

# how to run 2sls in stata?

The first step for running 2SLS is to ‘distribute the roles’, that is, to determine which variables are playing the role of the outcome variable (Y), the endogenous variable(s) of interest (X), the other exogenous RHS variables (W-s) and the instruments (Z-s). You are strongly advised to make this ‘accounting step’ before you start typing.

The command for 2SLS is ‘ivregress’, and the first argument is to specify that you want to use the 2SLS estimator (there are other estimators out there for IV regression):

ivregress 2sls

The next argument is the outcome variable’s name:

ivregress 2sls y

Then come the ‘other exogenous variables on the right-hand side’, now let us assume we only have 3 of them: w1, w2, w3

ivregress 2sls y w1 w2 w3

Now you will specify the first stage by giving the endogenous variable of interests (x1, x2) and the instruments (z1, z2) in parentheses, divided by an equality sign and of course without any commas (remember: every Stata line contains only 1 comma, and it has the role of dividing the arguments of a command from the modifying options):

ivregress 2sls y w1 w2 w3 (x1 x2 = z1 z2)

This code would already run, but for this course it is also compulsory to use the White standard errors that are robust to heteroskedasticity – this is the ‘robust’ *option*, just like with the ‘regress’ command. This is how we get our syntax:

ivregress 2sls y w1 w2 w3 (x1 x2 = z1 z2), robust

# Implementing The hausman-test in stata

The Hausman-test tests the null hypothesis that the 2SLS coefficients are the same as the OLS coefficients. Some authors (somewhat mistakenly) write that : ‘X is exogenous’, and : ‘X is endogenous’. It is fine if you do the same. One thing to remember about the version of the Hausman-test implemented in Stata that it requires homoscedasticity in the OLS regression. (Oh, well. It turns out in practice this is not terrible, and there are generalizations.)

You need to be able to interpret the Stata output of the Hausman-test, but do not have to remember the codes below (except for 1 subsection of the homework).

You need 5 steps, corresponding to 5 lines to implement the test:

1. Run the OLS regression WITHOUT the ‘,robust’

regress y x1 x2 w1 w2 w3

1. Store the estimates of this regression under some name (I will have the name ‘OLS’)

estimate store OLS

1. Run the 2SLS regression WITHOUT the ‘,robust’

ivregress y w1 w2 w3 (x1 x2 = z1 z2)

1. Store the estimates of this regression under another name (I will choose ‘TSLS’)

estimate store TSLS

1. Run the Hausman-test by giving the command ‘hausman’ and putting the labels you saved your estimates under *in the right order* (!!) as arguments

hausman TSLS OLS