SGPE Econometrics Lab 2: Stata Do Files and Log Files
Mark Schaffer
version of 23.09.2012

Introduction

This lab introduces you to Stata "do files" and "log files".

The Big Picture: **replicability** is an essential part of empirical science. If experimental results can't be replicated by other researchers, then as far as the rest of the research community is concerned, they might as well not have happened in the first place. The same principle holds in empirical work in economics. At the very minimum, for a given dataset, you must be able to replicate your original results.

This is good practice for another reason: you always can't realistically expect to get the answer to an empirical research problem in a single computer session. Empirical research is often long and involved. In practice, researchers gradually accumulate and modify the code that processes their data and generates their results.

A Stata **do file** is a text file with Stata **commands**. Stata will execute these commands in sequence, or selected sets of commands. The name of the text file ends with the extension "do", e.g., "mywork.do".

A Stata **log file** is a file with Stata **output**. Stata can write log files in more than one format, but we will always work with simple text output. Typically, you will write a do file that opens a log file, executes Stata commands, and then closes the log file. The name of the log file typically ends with the extension "log".

This lab will also show you more about Stata estimation commands, and about the `regress` command in particular. Stata commands leave behind saved results, and you will see how to display and work with them.

Preliminaries

This week we will again work with the data from Mankiw, Romer and Weil (1992), "A Contribution to the Empirics of Economic Growth".

The dataset has 12 variables:

`c_index`	numeric codes from 1-121 assigned to the countries
`c_name`	full country names
`c_code`	three-letter country codes (used for labelling graphs)
`cont`	continent
`nonoil`	dummy; =1 if the country does *not* export oil
`inter`	dummy; =1 if the country has good-quality data
`oecd`	dummy; =1 if the country was in the OECD
`gdp60`	GDP per member of the working age population in 1960
`gdp85`	GDP per member of the working age population in 1985

pop	average population growth over 1960-1985
igdp	average investment rate over 1960-1985
school	average secondary school enrolment of the working population over the years 1960-1985

You will often be working from the University network, and so you need a folder in which to save your do files, log files, etc. Last week you should have created a folder on the M: drive called "Econometrics" and a subfolder within that called "Lab1" (no spaces).

This week's work will be saved in (surprise!) a subfolder called "Lab2". Create this subfolder and copy the file "mrw1992.dta" into it. Do this in Windows.

To confirm you've done this correctly, open Stata and type the following:

```
cd M:\Econometrics\Lab2
```

This tells Stata to change the "working directory" to the "Lab2" subfolder in the "Econometrics" folder on the M: drive. From now on, when you load or save a do file, log file or dataset, it will look there or go there by default unless you actively specify another folder.

Next, type the following:

```
dir
```

"dir" stands for "directory", another term for "folder". This command says to list the contents of the current folder. The current folder is **M:\Econometrics\Lab2** so the contents should be the MRW dataset. You should see something like the following:

```
<dir>   9/25/12 10:50  .
<dir>   9/25/12 10:50  ..
 8.6k   9/21/12 18:24  mrw1992.dta
```

(In case you are curious, "." is shorthand for this folder; ".." is the folder above it in the directory tree structure.)

Do-file basics

Do files are simple text files. You can edit them using any text editor, but the simplest thing to do is to use Stata's built in do-file editor. The main reason is that it is very easy to execute the contents of do files from within the do-file editor.

Open the do-file editor by clicking on the do-file editor icon. There are two icons at the top with what look like little pencils on them; the one with the little black triangle is the do-file editor.

Enter the following into the do file:

```
cd M:\Econometrics\Lab2
use mrw1992, clear
```

```
sum
```

The first line automates the change of folder so that executing the do file always puts you in the right place. You'll recognize the next two commands from the previous lab: line 2 loads the MRW dataset, and line 3 summarizes the data.

Tip: The "clear" option tells Stata to load the data even if you have a dataset already in memory to which you have made changes. Almost always you will want to use this option when loading a dataset in a do file.

Now save the do file in your M:\Econometrics\Lab2 folder in the usual way (File > Save As).

Finally, execute the do file. You can do this from the do-file menu (Tools > Execute (do)), but the easiest way is to click on the icon on the right that looks like a piece of paper with writing on it.

Congratulations – you've just written and run your first do file.

Task 1: Write a do file that repeats parts of Tasks 6 and 7 from Lab 1

First, keep only the observations on countries where oil is not a dominant industry. We do this, as in the previous lab, by using the **drop** command. Add this after the line with the **sum** command:

```
drop if nonoil==0
```

(NB: we could just as easily have used the **keep** command: **keep if nonoil==1**.) Remember: in Stata the double **==** is the logical "equals" operator; the single **=** is the assignment operator.

If you ever make a major change to a dataset like this, it's a good idea to show what the consequences are. Do this by again summarizing the dataset:

```
sum
```

Next, add commands that create new variables that are the logs of GDP. Do this using the **generate** command:

```
gen lngdp60 = ln(gdp60)
gen lngdp85 = ln(gdp85)
```

Add a line after the variables are generated that tell Stata to summarize the new variables. Use the **sum** command for this, including the **detail** option so you can see the median, percentiles, etc.

```
sum lngdp60 lngdp85, detail
```

Save the do file and then execute it to confirm that it works correctly.

Next, create a variable with the average log growth rate over the 25 year sample period. Also label the variable. Put this in your do file:

```
gen growth = (lngdp85 - lngdp60) / 25
label variable growth "average growth rate"
```

Add a command to estimate a model using OLS, where the dependent variable is the growth rate and the independent variable is the log of GDP per worker in 1960 (see Lab 1). Use the **regress** command.

```
reg growth lngdp60
```

Finally, estimate an extended model with three additional explanatory variables: average population growth, the investment share of GDP in percent (i.e., the average of I/Y for the entire 1960-85 period), and secondary school enrolment (again averaged for the 1960-85 period):

```
reg growth lngdp60 pop igdp school
```

Save the do file and execute it.

The contents of this do file resemble in minature the contents of a do file that you would use for an Econometrics assignment or for something more involved: (a) load a dataset; (b) manipulate the data; (c) do some estimations and display the results.

Remark: Note that we do **not** save the dataset with the new variables. In practice, you will always save a datataset **with a new name**, and you will do this only when the data manipulations are very involved and the dataset is large, because in that case it saves some processing time. The reason we don't get into the habit of saving data is because it's usually unnecessary, and because the risk of accidentally zapping the original dataset is a real one. **Never modify your raw data!** (Remember – we want replicability.)

Task 2: Understanding the output of Stata's `regress` command

In this task we consider in detail the output generated by Stata's **regress** command and how to interpret it.

Start by estimating the basic bivariate regression in your do file:

```
reg growth lngdp60
```

You can do this by selecting the line and then clicking the "do" icon. You should obtain the following output:

```
. reg growth lngdp60

      Source |       SS       df       MS              Number of obs =      98
-------------+------------------------------           F(  1,    96) =    3.61
       Model |  .001121678     1  .001121678           Prob > F      =  0.0603
    Residual |    .0298044    96  .000310462           R-squared     =  0.0363
-------------+------------------------------           Adj R-squared =  0.0262
       Total |  .030926078    97  .000318826           Root MSE      =  .01762

------------------------------------------------------------------------------
      growth |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
     lngdp60 |   .0037724   .0019847     1.90   0.060    -.0001671     .007712
       _cons |  -.0106631   .0151842    -0.70   0.484    -.0408035    .0194773
------------------------------------------------------------------------------
```

The output comes in three parts: an ANOVA (analysis of variance) table in the upper left-hand corner, some basic statistics in the upper right-hand corner, and a table of regression coefficients and associated statistics as the main output. We consider these in reverse order.

Table of regression coefficients:

The table of regression coefficients reports, for each regressor, the estimated coefficient, the standard error, the associated t-statistic and p-value, and a 95% confidence interval:

```
      growth |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
     lngdp60 |   .0037724   .0019847     1.90   0.060    -.0001671     .007712
       _cons |  -.0106631   .0151842    -0.70   0.484    -.0408035    .0194773
------------------------------------------------------------------------------
```

How do we interpret the entries for the **lngdp60** row?

The estimated coefficient is **.0037724**. What does this mean? The dependent variable is **growth**, which is the average annual log growth rate (not the percentage growth rate). The explanatory variable is **lngdp60**, which is the log of the value of GDP per worker at the start of the period, 1960. To fix magnitudes, here is a summary of these two variables:

```
. sum growth lngdp60

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
      growth |        98    .0179996    .0178557  -.0270272   .0663693
     lngdp60 |        98    7.597921    .9014184   5.948035   9.422382
```

So a mean of 0.018 for **growth** means that the average annual growth rate for this period was 1.8% per annum.

Now consider the estimated coefficient. The value is approximately 0.0038. If **lngdp60** increased by 1 unit, then **growth** would increase by 0.0038. Consider these magnitudes in turn. A 1-unit increase in **lngdp60** is a very big increase in GDP per worker. To be precise, since $e^1=e=2.72$ (we are raising e to the power 1 because we are considering a 1-unit increase in a natural log), it corresponds to a 100*(2.72-1)=172% increase in GDP per worker, i.e., between doubling (100% increase) and tripling (200% increase). The change in the growth rate is 1*0.0038 = 0.0038 = 0.38%. We can say that "if GDP per worker in 1960 goes up by 172%, the annual growth rate goes up by 0.38 percentage points".

Note: if you are working with a RHS variable that is a log, you might find it convenient to think in terms of a doubling in the underlying variable. Since *ln(2)*=0.69, and 0.69*0.0038 = 0.0026, we can say "if GDP per worker in 1960 doubles, the annual growth rate goes up by 0.26 percentage points".

So much for interpretation. How *precise* is the estimate? This is what the rest of the line tells us about.

Probably the easiest way to think about the precision of the coefficient estimate is in terms of the confidence interval. The last two figures in the row are the endpoints of a 95% confidence interval: **[-0.0001671 , 0.007712]**. Our model is

$$\text{growth} = \alpha + \beta * \text{lngdp60}$$

β is a population parameter, i.e., we don't know what it is. Intuitively, and very roughly speaking, we are "95% confident" that the true value of β lies in the interval generated by our OLS estimation: **[-0.0001671 , 0.007712]**.

This is a rather wide confidence interval. Say we were still interested in the impact of a 1-unit increase in **`lngdp60`**. We are "95% confident" that the impact of this is somewhere between a fall in the annual growth rate of -0.017 percentage points, and an increase in the annual growth rate of 0.78 percentage points.

The columns for the standard error, t-statistic and p-value have essentially the same information about the precision of the estimated parameter as the confidence interval, just presented differently. The null hypothesis being tested is:

H_0: $\beta=0$

The OLS estimate of the parameter, $\hat{\beta}$, is a statistic, and as such it has a variance. The square root of the estimated variance of $\hat{\beta}$ is called its "standard error". Under the null hypothesis above, the ratio $\hat{\beta}/SE(\hat{\beta})$ has a t-distribution with N-K degrees of freedom, where K is the number of regressors including the constant (here, K=2). This ratio is our "test statistic". The entry in the "t" column in the regressor table produced by Stata is simply this ratio. Finally, if the null hypothesis is correct – i.e., β really is zero – we know this test statistic comes from a t-distribution. This means we can calculate the probability that we would observe a test statistic with at least this value, or the "p-value". By convention, we normally calculate this for alternatives in which the true β is either very positive or very negative, i.e., we usually work with 2-tailed tests. Stata reports, by default, the p-value using a 5% significance level.

A simple and intuitive way to link the t-test and the confidence interval is to consider whether 0 appears in the confidence interval. If 0 **is not** in the 95% confidence interval, then the reported p-value will be **less** than 5%. If 0 **is** in the 95% confidence interval, then the reported p-value will be **greater** than 5%.

Since the critical value for a 2-tailed test at the 5% significance level using the t-distribution with (N-K) = (98-2) = 96 degrees of freedom is 1.985, the confidence interval being reported by Stata is $\hat{\beta} \pm 1.985 * SE(\hat{\beta})$.

The t-distribution is almost indistinguishable from the normal as soon as the degrees of freedom are more than 30 or so, so in practice you will usually get virtually the same results if you assume $\hat{\beta}/SE(\hat{\beta})$ has a normal distribution. And since the critical value for a 2-tailed test using the Normal is 1.96, you can usually work out an approximate confidence interval by adding/subtracting 2*$SE(\hat{\beta})$ to the estimated coefficient. Similarly, you can usually conduct an approximate test of the null hypothesis that β=0 by seeing whether the ratio $\hat{\beta}/SE(\hat{\beta})$ is greater than 2 or not.

The table of regression coefficients includes as the last row all the usual output for a variable called **_cons**. This is just the output for the estimate of the intercept α. Stata always adds a constant unless you specifically say not to with the **nocons** option.

One useful way to think about this is that α is the coefficient on a variable that is just a column of ones. You can demonstrate this by creating such a variable and using it as a regressor, but also remembering to exclude the constant that Stata automatically inserts. Add the following to your do file after the estimate reported for the simple bivariate regression:

```
gen one = 1
reg growth one lngdp60, nocons
```

Execute these lines and compare the results to the bivariate regression with Stata's automatically-included constant:

```
reg growth lngdp60
```

The main output is identical. The statistics in the top half are not, however. The reason will be clear below.

How do we interpret the constant? One way to think of it is that it is an estimate of the conditional mean of the dependent variable, in this case **growth**. But it is an estimate that is usually not calibrated in a way that is particularly useful. The estimated constant answers the question: "Say we know the log level of GDP per worker in 1960. What is our best estimate of the growth rate if the log level of GDP per worker were zero?"

This is an answer to a not-particularly-useful question. On the other hand, the answer to the following question is even less useful, because we know in advance what the answer is: "Say we know the log level of GDP per worker in 1960. What is our best estimate of the growth rate if the log level of GDP per worker were the average for the sample?"

What is the answer to this question? Add the following lines to your do file:

```
sum growth lngdp60
gen lngdp60_c = lngdp60 - 7.597921
sum growth lngdp60 lngdp60_c
reg growth lngdp60_c
```

Note the value of the new estimated constant. It is, in fact, exactly the same as the mean of **growth**. Question: why? [Short answer: the geometry of OLS guarantees that the regression line passes through the sample mean.]

Basic statistics in the upper right-hand corner

The statistics reported in the upper-right hand corner of the regression for our simple bivariate model were:

```
Number of obs =      98
F(  1,    96) =    3.61
Prob > F      =  0.0603
R-squared     =  0.0363
Adj R-squared =  0.0262
Root MSE      =  .01762
```

We consider these in turn.

The number of observations is the number used in the regression, not the number in the entire dataset.

The F-test is a test of the joint significance of all K regressors **not counting the constant**. The reason we exclude the constant is by convention; the F-test is a test of the "model", and we don't consider whether or not the dependent variable has a non-zero mean as an important component of our model. In this case, K=2, so the F test has K-1=1 numerator degree of freedom. The denominator degrees of freedom = N-K = 98-2 = 96.

Note that, for the bivariate regression **only**, the t-test of **`lngdp60`** and the F-test in the upper right-hand corner are essentially the same test: t^2=F, i.e., 1.90^2=3.61. The p-values are identical.

R^2 is a measure of "goodness of fit", i.e., how well the model fits the data.
Basic model:

$$\text{growth}=\alpha+\beta*\text{lngdp60}$$

Basic model including subscripts and the error term:

$$\text{growth}_i=\alpha+\beta*\text{lngdp60}_i+\varepsilon_i$$

i.e., the error term is everything that's not in the model.

Estimated model with fitted values of the dependent variable:

$$\widehat{\text{growth}}_i=\hat{\alpha}+\hat{\beta}*\text{lngdp60}_i$$

Residuals defined:

$$\hat{\varepsilon}_i=\text{growth}_i-\widehat{\text{growth}}_i$$

i.e., the residuals are the differences between the actual dependent variable and what the model predicts for it based on the actual values of the regressors.

The definition of R^2 is the share of total variation in the dependent variable explained by the model. It can be calculated using either the fitted values of the dependent variable or using the residuals:

$$R^2 = \frac{\sum_i (\widehat{\text{growth}}_i - \overline{\text{growth}})^2}{\sum_i (\text{growth}_i - \overline{\text{growth}})^2} = 1 - \frac{\sum_i \hat{\varepsilon}_i^2}{\sum_i (\text{growth}_i - \overline{\text{growth}})^2}$$

The traditional R^2 reported in regression output is a "centred R^2". In other words, we "demean" the dependent variable in the formulae above. The mean corresponds to the presence of a constant term in the model, i.e., the parameter α. If our model did not have a constant term, Stata would report an "uncentred R^2", i.e., without the means removed. This is why the R^2 when we include a constant,

```
reg growth lngdp60
```

is different from when we don't:

```
reg growth one lngdp60, nocons
```

In the former, Stata is reporting a traditional "centred R^2"; in the latter, it is reporting an "uncentred R^2". This is also why the F-statistics reported in the two regressions are different. In the former, Stata is not counting the constant as part of the model, and so the F test has 1 numerator degree of freedom, **`F(1,96)`**. In the latter, Stata is being told that there is no constant, and that the model has **two** regressors, so it reports a test of their **joint significance**, and the F test has 2 numerator degrees of freedom, **`F(2,96)`**.

The "adjusted R^2" is an adjustment to the traditional R^2; we don't use it much in econometrics, and we don't discuss it here.

Finally, "Root MSE" stands for "Root Mean Squared Error":

$$RMSE = sqrt\left(\frac{1}{N-K}\sum_i \hat{\varepsilon}_i^2\right)$$

In other words, the RMSE is the square root of the "average" squared residual, except that instead of dividing by N to get the average, we divide by N-K, where K=number of estimated parameters (in this case, 2). The RMSE is an unbiased and consistent estimate of σ, the square root of the variance of the error ε_i.

Question: is there any difference *asymptotically* between the traditional formula for estimating σ above, and the following simplified formula? And why is "large" a good name for this estimate of σ?

$$\hat{\sigma}_{large} = sqrt\left(\frac{1}{N}\sum_i \hat{\varepsilon}_i^2\right)$$

Question: how is this related to using t-statistics for testing the significance of coefficients and using the Normal approximation instead?

Question: in practice, will it usually matter much if we use traditional formula for estimating σ or the simpler "large sample" formula above?

ANOVA output in the upper left-hand corner

Stata's **regress** also reports an analysis-of-variance or ANOVA in the upper left-hand corner of the output:

```
      Source |       SS       df       MS
-------------+------------------------------
       Model |  .001121678     1  .001121678
    Residual |    .0298044    96  .000310462
-------------+------------------------------
       Total |  .030926078    97  .000318826
```

A full discussion of ANOVA is a major topic in itself. Here we just relate the output to quantities and concepts you have already seen.

SS = "sum of squares"
df = "degrees of freedom"
MS = "mean square"

The entries in the SS column all assume the presence of a constant in the model, unless you have specified the **nocons** option:

$$\text{Total SS} = \sum_i \left(\text{growth}_i - \overline{\text{growth}}\right)^2$$

$$\text{Model SS} = \sum_i \left(\widehat{\text{growth}}_i - \overline{\text{growth}}\right)^2$$

$$\text{Residual SS} = \sum_i \left(\text{growth}_i - \widehat{\text{growth}}_i\right)^2 = \sum_i \hat{\varepsilon}_i^2$$

and therefore you could calculate the R^2 for the regression using the output on the LHS: R^2 = Model SS / Total SS = 0.32872148/0.62676543 = 0.5245, which is what is reported by Stata on the RHS.

The degrees-of-freedom column reports:

The total degrees of freedom available = N-1. (Why subtract 1? By convention, although we estimate a constant term we don't consider it "part of the model". But we have to estimate it anyway.)

The degrees of freedom used up by the model parameters = K-1. (Why subtract 1? Same reason as above: by convention, the constant term is not "part of the model".) In this case, K=2 because there are estimates of two parameters, α and β, so there are K-1 = 2-1 = 1 model degree of freedom.

The "residual" or remaining degrees of freedom = (N–K) = 98–2 = 96.

Note that the degrees of freedom for "Model" and "Residual" are the same as those reported for the F-test of the model in the upper RH output.

The MS or "mean square" column reports the SS column divided by the degrees-of-freedom column. Note that the square root of the figure reported as the MS of the residual, sqrt(0.000310462), is exactly what is reported as the "Root MSE" in the RHS output.

Task 3: Add comments to the do file

It is very frustrating to return to your code after a break and waste time trying to figure out what it did. It is even more frustrating for someone else to try to work out what your code did based only on the commands.

The solution to this is to *add comments to your do file*. There are various ways to add comments (have a look at Stata's online help – type **`help comment`**). The simplest way is the following:

> Start a line in the do file with a **`*`**. Be sure the **`*`** is the very first character on the line. Everything after the **`*`** will be ignored by Stata.

Practice this by adding comments to your do file that explain what the different blocks of code do, e.g.,

```
* Change to working directory.

* Load the MRW dataset.
* Use the clear option in case memory already has data in it.

* Summarize the dataset.

* Generate new variables.

* Estimate a simple bivariate model

* Estimate a model including population, I/Y and schooling
* as explanatory variables.
```

Note that the simplest way to make long lines readable is to split them up.

After adding the comments, save and execute the do file to confirm it works.

Note: in recent versions of Stata, you can use a double-forward-slash **`//`** instead of a **`*`** to start a line with comments.

Task 4: Create a log file

A "log" file is a record of what Stata reports in the results window. We will almost always work with log files in simple text format. These can be edited with any text editor.

You can start a log file interactively, but in practice you will usually do this in your do file. Running your do file will then create a log of the output it generates, and you can then inspect/print/submit/etc. it.

Add the following lines to very beginning of your do file:

```
log using M:\Econometrics\Lab2.txt, text replace
```

This line tells Stata to start saving output to a file called "Lab2.txt". The **text** option guarantees that it is a simple text (ascii) file. The **replace** option means that if a previous log file called "Lab2.txt" exists, it will get deleted and replaced with this new one.

Tip: If you want to add new output to the bottom of your log file, instead of replacing it entirely, use the **append** option instead of the **replace** option.

Next, add the following line to the bottom of your do file:

```
log close
```

Save and execute the entire log file. Now go to Windows explorer and navigate to your M:\Econometrics\Lab2 folder. In it will be a file called "Lab2.txt". Open it using a text editor and you will see your Stata output.

Tip: To save yourself some hassle, add the following line to the top of your do file, *before* the "log using" command:

```
capture log close
```

Explanation: This line tells Stata to close any log files that are open, i.e., if you are already writing output to a log file, Stata will close it. This makes sense – almost always you will want to write your output to a single log file, and you don't want stuff going simultaneously into multiple log files.

What does **capture** do? It covers the situation where *you don't actually have any log files open*. If you tried to close a log file, but it wasn't open to begin with, Stata would exit with an error. **capture** is a very convenient trick command: **capture [command]** tells Stata, "Execute [command], but don't complain if it doesn't work." (If this is confusing, don't worry – just get into the habit of using this line.)

Task 5: Understanding the output of Stata's `regress` command (repeat)

In this task you should repeat all of the examination of Stata's output as in Task 2, but for the expanded model:

```
reg growth lngdp60 pop igdp school
```

Things you should consider and discuss amongst yourselves:

- How do you interpret the sign and size of the coefficient on the main regressor of interest, **lngdp60**?
- What happened to the standard error for the coefficient on **lngdp60** vs. the simple bivariate model? What happened to the t-stat and confidence interval?
- How do you interpret the F-test in the upper RHS output?
- What happened to the R^2 compared to the simple bivariate model? What happened to the RMSE?
- Why does the F-test in the upper RHS output have the reported number of degrees of freedom?
- How do you interpret the sign, size and significance of the coefficient on **pop**?
- How do you interpret the sign, size and significance of the coefficient on **igdp**?
- How do you interpret the sign, size and significance of the coefficient on **school**?

Put notes about your discussion as comments in your do file. Run the do file so this appears in the log file as well.

Often it is handy to be able to compare the output of different regressions side-by-side. In Stata we do this by first saving the regression results under different names, and then combining the results into one table.

Regression results are stored in Stata's memory by using the **estimates store** command or **est store** for short. (Note: this does not save them permanently to your working folder; if you exit Stata, the results are gone.) You should store the results **immediately** after the estimation (since running another regression will wipe the preceding estimates from Stata's memory). We'll call the two models "basic" and "expanded". Modify your do file so that immediately after the estimations you store the results with these names:

```
reg growth lngdp60
est store basic
```

and

```
reg growth lngdp60 pop igdp school
est store expanded
```

Now that you've done this, you can combine the results using **estimates table**:

```
est table basic expanded
```

The result is very basic: just the coefficients are reported:

```
-------------------------------------
    Variable |   basic       expanded
-------------+-----------------------
     lngdp60 |  .00377244   -.01099848
         pop |              -.00091787
       igdp |               .00130072
      school |               .00254465
       _cons |  -.0106631    .06686524
-------------------------------------
```

The **est table** command has various formatting options. Usually we will want standard errors as well:

```
est table basic expanded,  b se
```

Or we could report stars next to the coefficients, and while we're at it, format the coefficient estimates so that we allocate a fixed 7 spaces to each (including the decimal point and the negative sign, if any), and a fixed 4 spaces to follow the decimal point):

```
est table basic expanded,  b(%7.4f) star
```

The output now looks like this:

```
-------------------------------------
    Variable |   basic       expanded
-------------+-----------------------
     lngdp60 |  0.0038      -0.0110***
         pop |              -0.0009
       igdp |               0.0013***
      school |               0.0025***
       _cons | -0.0107       0.0669***
-------------------------------------
legend: * p<0.05; ** p<0.01; *** p<0.001
```

Add this **est table** command to your do file and run it.

Tip: Stata's built-in **est table** has various limitations. One of Stata's strongest features is the large number of user-written extensions to Stata. The package **estout** by Ben Jann has a command **esttab** with a number of options that Stata's official command does not, including the ability to output tables to RTF format (i.e., Word-compatible) files. To find it and install it, just type

```
findit estout
```

and follow the instructions – a few clicks – to install.

Task 6: Display and work with saved estimation results and global macros

Stata's estimation and other commands usually save additional statistics, matrices, etc. that are not in the main display of results. **regress** is an "eclass" command ("e" for "estimation"), so you can display these results using the command **ereturn**.

Do the following from *within* Stata, i.e., type it into the command window and *not* in your do file:

```
reg growth lngdp60
ereturn list
```

The latter command lists all the saved estimation results.

You can also list selected saved results. For example,

```
ereturn list N
ereturn list r2
```

To do calculations with these saved results, you can use the Stata **display** command. This turns Stata into a glorified hand calculator. For example, say we want to calculate the value of R^2. Note that the saved regression results include **e(mss)** and **e(rss)**. MSS="model sum of squares" and RSS="residual sum of squares", so by definition TSS="total sum of squares"=MSS+RSS. To calculate R^2 by hand, type the following into the command window (**di** is the abbreviation for the **display** command):

```
di e(mss) / (e(mss)+e(rss))
```

Now compare to the reported value of R^2:

```
di e(r2)
```

They should be the same. (Remember, Stata syntax is case sensitive; **di e(R2)** won't work.)

Saved eclass results are available until the next estimation command; run another regression and the old results disappear. To save specific numbers for later use, we use "macros".

A global macro is a name which has a number or text assigned to it. When you put a $ in front of the name, Stata knows it is a macro and looks up the value. Here is how to calculate R^2 using global macros:

```
global MSS = e(mss)
global RSS = e(rss)
global TSS = $MSS + $RSS
global Rsq = $MSS/$TSS
di $Rsq
```

Add these lines to your do file after the estimation of the extended model (with **pop**, **igdp** and **school** as explanatory variables), but *before* the **capture log close** command (otherwise it won't appear in your log file). Also add comments explaining what the code does. Run just the lines executing the model and this calculation (highlight and then click the do icon). When you've confirmed the code works, save the do file and execute it. This calculation will now appear in your log file.

Note that **est table** can also access these saved statistics. For example, say you want the R^2 reported along with the coefficients, SEs and stars. Use the **stats** option:

```
est table basic expanded,  b(%7.4f) star stats(r2)
```

Add this to your do file.

Task 7: Calculate and work with residuals and predicted values

Stata's estimation commands come with "postestimation" commands of various sorts. These can do certain calculations, perform certain tests, etc.

A very common postestimation commend is the **predict** command. After linear regression, predict can be used to calculate predicted values and residuals. The predict command is like the gen command in that it creates a new variable with a name of your choice. We'll call our residuals "ehat".

Here is how to calculate residuals after the regress command. First, estimate the simple bivariate version of the model. Then type this into the command window (don't forget the comma!):

```
predict double ehat , resid
```

Note the use of **double**. This means Stata should use "double precision", i.e, the new variable ehat should be calculated as precisely as possible. When you work with residuals, it is often in order to calculate a test statistic, and you will want to use the most accurate calculation possible.

Here is how to calculate fitted values:

```
predict double yhat , xb
```

If we square our residuals and then sum them, it should match the saved result **e(rss)** after the estimation. Create a new variable called ehatsq. Also make it double precision:

```
gen double ehatsq = ehat^2
```

Now summarize it:

```
sum ehatsq
```

Where is the sum of the residuals? It's not displayed (even if you use the detail option), but it *is* a saved result. **summarize** is an "rclass" command, which means that saved results can be displayed using **return** (not **ereturn**! – that's for eclass commands):

```
return list
```

And you will see the sum saved as **r(sum)**.

If you want to display just the saved sum, you have two options:

```
return list sum
```

or

```
di r(sum)
```

And compare this to the RSS you saved as a macro:

```
di $RSS
```

Add the above to the end of your do file, after the section with the macros (and again, *before* the **capture log close** command). Also add comments. Save the do file and execute it.

Tip: What if ehat already exists as a variable when you try to create it using predict? Stata will not overwrite the existing variable; instead, it will exit with an error. You could make sure that the variable is dropped by using the **drop** command:

```
drop ehat
```

But what if the variable ehat does *not* exist when you try to drop it? Then Stata will exit with an error again.

Solution: the **capture** command. In your do file, when creating a new variable, precede the command with **capture drop [name of variable]**. This tells Stata, "drop the variable if it exists, do nothing if it doesn't". For example, in your do file above, your use of the **predict** command would be

```
capture drop ehat
predict double ehat , resid
```

And

```
capture drop yhat
predict double yhat , xb
```

And

```
capture drop ehatsq
gen double ehatsq = ehat^2
```

Add the **capture** lines to your do file. Also add comments. Save the do file, execute it and confirm that it works.

Task 8: Plot a regression line and save the graph

In last week's lab, we looked at a scatterplot of **growth** vs. **lngdp60**. We also combined the scatterplot with a plot of the regression line. Here we show how to save the graph in a format that can be imported into other packages.

Recall from last week that you do a simple scatterplot using the **scatter** command:

```
scatter growth lngdp60
```

And you can plot the linear regression line using the **lfit** subcommand of the **twoway** graph command:

```
twoway lfit growth lngdp60
```

And since **scatter** is also a **twoway** subcommand, you can combine these two very easily:

```
twoway (scatter growth lngdp60) (lfit growth lngdp60)
```

Often, we want to save graphs for later use. Stata has its own **gph** format for storing graphics files. Such a file will have a .gph extension. Unfortunately, most other software packages don't understand this format. Instead, what we need to do is use the **graph export** command. The default action of **graph export** is to take whatever graph we are looking at and export it. This is a little risky, because we might have several graphs open and accidentally export the wrong one, but if we do it in a do file we won't run into this problem.

To export the graph to a file in WMF (Windows metafile) format called "convergence.wmf", change your do file so it reads:

```
twoway (scatter growth lngdp60) (lfit growth lngdp60)
graph export convergence.wmf
```

Other available formats include TIFF, PostScript, Windows Enhanced Metafile, etc. If you want to export to Windows Enhanced Metafile Format instead, just change the line to:

```
graph export convergence.emf
```

After adding these lines to your do file, save the do file and execute it in its entirety.