Econ 452 Section 5 - STATA

Connor Cole

October 18, 2015

Using STATA as a Calculator

If you would like to use STATA as a calculator for scalars, you can use the `display` command. Just prefix some calculation you would like to run using the `display` command:

```
display 3*4 / .5
```

Regression, Continued

Post-Estimation Regression Predictions

After running a regression, you can create an additional variable for each observation recording either $\hat{y}$, or the predicted value of the dependent variable for a given individual, or $e = y - \hat{y}$, the unexplained residual between the predicted value of the dependent variable and the observed value.

To predict $\hat{y}$ for each observation and save it in an additional variable, we type:

```
predict NewVariableName, xb
```

To predict $\hat{e} = y - \hat{y}$ for each observation and save it in an additional variable, we type:

```
predict NewVariableName, resid
```

Note that these command must follow immediately from a regression.

Post-Estimation Regression Predictions at a Given Point

The previous method will create predictions for observations in the dataset. However, sometimes, we might want to use the output from a regression to predict the dependent variable for a specific

set of covariates, and we might be interested in the standard error associated with this prediction from the standard errors in our estimation of the coefficients of the regression. Let's say that we've run a single variate regression:

```
regress VariableName1 VariableName2
```

If we wanted to predict the value of `VariableName1` when `VariableName2` equals 10, we would type:

```
margins, at(VariableName2=10)
```

Consider the image below offering output from running this `margins` command:

Figure 1: Margins

```
. margins, at(VariableName2=10)

Adjusted predictions                            Number of obs     =      6,828
Model VCE    : OLS

Expression   : Linear prediction, predict()
at           : VariableNa~2    =          10
```

	Delta-method					
	Margin	Std. Err.	t	P>|t|	[95% Conf.	Interval]
_cons	**1.374218**	**.0211147**	**65.08**	**0.000**	**1.332826**	**1.415609**

Similar to the STATA output we have seen before, the first few lines of this output just reiterates the command we have given STATA, including the value of the variables we assumed for out predictions. Underneath this line is a table stating the predicted value of the dependent variable, its associated standard error (reflecting the fact that the coefficients of the model we are using for our prediction have been estimated and each have their own standard error), and a t-test for the predicted value being different than 0, and a 95% confidence interval.

The `at` command we have used is quite flexible and allows us to estimate predicted values conditional on statistics of our covariates. If we wanted to predict the value of `VariableName1` at the 75th percentile of `VariableName2`, we would type:

```
margins, at((p75) VariableName2)
```

If we wanted to predict the value of `VariableName1` at the mean of `VariableName2`, we would type:

```
margins, at((mean) VariableName2)
```

Note that if we run a multivariate regression, we can proceed similarly by writing:

```
regress VariableName1 VariableName2 VariableName3 VariableName4

margins, at((p75) VariableName2 VariableName3=2 (mean) VariableName4)
```

An Aside on Stored Values in STATA

NOTE: This section is completely optional and only present here because it provides tools that are in general useful for STATA programming. While these commands covered in this section might help with problem sets, they aren't necessary, and are simply described here for those who might be interested and for future reference.

Whenever STATA reports output from some command, such as `summarize` or `regress`, the output is stored in scalars and matricies, along with additional outcomes not directly reported in STATA. For example, when we use the `regress` command, STATA directly reports standard errors for estimated regression coefficients, and does not report covariances between regression coefficients. However, those covariances are stored by STATA after it estimates the regression. Sometimes you may want to report these values or compute functions of them.

There are two different 'classes' of commands in STATA and, correspondingly, two different ways STATA saves data after running a command. Commands like the `summarize` or `margins` command are seen by STATA as commands that 'report' certain values (technically, an r-class command) while commands like the `regress` command are seen as commands that 'estimate' values (technically, an e-class command). On a practical level, this means that calling the results from the `summarize` and `margins` command can be slightly different than calling results from the `regress` command.

You can access r-class and e-class returns specifically by typing `return list` and `ereturn list` respectively. For example, if we wanted to see these returns after using the `regress` command, we would type:

```
regress VariableName1 VariableName2

return list

ereturn list
```

Note that STATA saves information in three ways: as scalars, as matricies and as 'macros.' Scalars are simply numbers, and STATA directly reports the value of a scalar when we use the previous commands to display what values are saved by STATA. Matricies are too large to directly report, so when we type the commands above we just see the dimensions of the matrix, where the first number reports the number of rows and the second number reports the number of columns. Lastly, macros are text strings that report features of the command, such as the estimation process used and the variables considered.

As there are three different types of output saved after commands, there are three different ways of saving this information after we run a command. We're generally interested in saving this information in separate objects because each time we run these commands these individual objects are replaced with the new values from the new command, and saving them separately is the only way to ensure that individual output from previous estimation commands can be accessed later. Note that these methods save objects that are not variables that would appear if looked at the data with the `browse` command.

Saving Scalars

Let's say we ran the regression `regress VariableName1 VariableName2` and consider the scalars from these saved results. From the `ereturn list` command, we see that STATA has saved R^2 as scalar `e(r2)`. We can see the value of R^2 stored as a scalar by typing:

```
disp e(r2)
```

We can furthermore save this value in a new scalar and access it later by typing `scalar` to clarify that we would like to save a scalar, the name of the scalar we would like to create, and lastly the output from the regression we would like to save. In this case, we would type:

```
scalar rsquared = e(r2)
```

Then we can access this value by typing:

```
scalar list rsquared
```

We can calculate functions of this value, for example multiplying this value by 2, by typing:

```
disp rsquared*2
```

Saving Macros as Text Strings

Now, consider the macros saved by a regression. Running a regression saves a macro saying that we have run a linear regression under the `e(title` macro. We can see this macro by typing:

```
disp "`e(title)'"
```

We can save this macro as a text string and access it later by typing `local` to clarify that we would like to save a string of information locally, the name of the string we would like to create, and lastly the macro from the regression we would like to save. In this case, we would type:

```
local method = "`e(title)'"
```

We can then access this string and display it by typing:

```
disp "`method'"
```

Saving Matricies

Lastly, consider the matricies saved by a regression. From the `ereturn list` command, we see that regressions save a variance-covariance matrix of the estimated coefficients in our regression under the matrix `e(V)`. We can display this matrix by typing:

```
matrix list e(V)
```

We can save this matrix and access it later by typing `matrix` to clarify that we would like to save a matrix, the name of the matrix we would like to create, and lastly the matrix from the regression we would like to save. In this case, we would type:

```
matrix variancecovariance = e(V)
```

We can then access this matrix and see it by typing:

```
matrix list variancecovariance
```

If we wanted to access specific entries in the matrix, for example the cross correlation between the two different estimated parameters stored in the first row and second column of the matrix above, we would save the new object as a scalar and see it by typing:

```
scalar covariance = variancecovariance[1,2]
```

Then, with this additional `covariance` scalar, we could do the same operations we've considered before for scalars.

Creating New Variables with Scalars and Macros

As noted before, all of these objects are saved in ways that wouldn't appear in the dataset if we typed `browse`. Sometimes, we may want to save these objects as variables so that we can use the variable manipulation techniques we have already learned. For example, we might want to multiply some variable in our dataset by the R^2 value from a regression output, or we might want to create a string variable reporting the estimation process we have used.

We can directly create variables using the output from our regression by just using the e and r-class returns in a statement. For example, if we had run the regression described previously, and we wanted to create a new variable with a value for each observation reporting the value of some scalar output from the regression, say the R^2, we would type `generate`, the name of the new variable we would like to create, and lastly the name of the scalar we want to use. In this case, we would type:

```
generate rsquaredvar = e(r2)
```

If we had saved this variable as a scalar titled `rsquared`, we could also accomplish this goal by typing:

```
generate rsquaredvar = rsquared
```

If we want to save a new variable that records for each observation in the dataset the text string in some macro, say the fact that the estimation we have run in this case is a linear regression, we would type `generate`, then the name of the new variable we would like to create, and lastly the macro we would like to use. In this case, we would type:

```
generate estimationprocess = "`e(title)'"
```

Again, as in the scalar case, if we had saved this macro as a local string titled `method`, we could accomplish this goal by typing:

```
generate estimationprocess = "`method'"
```