The use of functions in MATLAB

Ralf Becker

1 Overview

Functions are an essential toolkit in every programming language. They are used to "outsource" a piece of code that is so generic that it may be reused on a number of occasions. For it to be able to be reused it is written such that the things that may change (i.e. different datasets) are treated in a way that makes them easy to change.
In fact, a good analogy is a drinks vending machine. The box or machine (or in our language, function) hides a large number of things (computer, mechanics etc.) from the eyes of the user. All the user does is to provide some input (money and choice of drink), then the machine does its stuff, and eventually delivers some output, hopefully an ice-cold can of your favourite softdrink. Here we will do exactly the same. We will write a bit of code that does something useful (in our case it will calculate an OLS regression). To do that it will require the user to provide some input. The function will do its work and deliver back some output.

2 Econometric Background

This is not the place to review the Econometric Theory in detail, but to make the context clearer consider that we are concerned with estimating a regression model

$$\mathbf{y} = \mathbf{X}\beta + \epsilon \tag{1}$$

where $\mathbf{y}$ is a $(n \times 1)$ vector that contains all n observations for the dependent variable and $\mathbf{X}$ is a $(n \times k)$ that contains all explanatory variables. The $(k \times 1)$ vector β represents the unobserved population coefficient vector and ϵ is a $(n \times 1)$ vector of unobserved error terms. The OLS estimator for the unknown parameter vector is of course

$$\widehat{\beta} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y} \tag{2}$$

You will also recall that useful associated statistics to such a regression are the standard errors

of $\widehat{\beta}$, the residual sum of squares and the R^2, all of which you can review in the Econometrics textbook of your choice.

3 Function structure

In general a function will look like this:

```
function [out1,out2,...]  = FunctionName(in1,in2,...)
programming commands;
...
programming commands;
end
```

We have a set of input variables (`in1, in2, etc.`) which will be used in a set of calculations (`programming commands`). These calculations will use the input variables to calculate some outputs (`out1, out2, etc.`). The function then hands back the values for these variables (`out1,out2,...`) such that they can be used later in any subsequent calculations.

So, before we continue we need to specify what the inputs ought to be and what outputs we should expect from our function. The input that is required to estimate a regression is the following:

- A vector that contains all observations for the dependent variable, $\mathbf{y}$
- A matrix, $\mathbf{X}$ that contains all explanatory variables in the columns. This matrix should have the same number of rows as $\mathbf{y}$.
- (optional) A variable that indicates whether we want the regression output printed into the MATLAB command window or not.

The function will then estimate a regression and deliver some output. It is of course in the hands of the programmer (that is you!) to determine what regression outputs you want. For the sake of this exercise we shall deliver the following

- $\widehat{\beta}$ (or `b` below), the vector containing the estimated regression coefficients.
- $s_{\widehat{\beta}}$ a $(k \times 1)$ vector with the estimated OLS standard errors for $\widehat{\beta}$ (`bse` below).
- $\widehat{\epsilon}$, the $(n \times 1)$ vector of estimated regression residuals (`res` below).
- n, the number of observations used.

- RSS, the residual sum of squares.
- R^2 (or below `r2`).

4 OLSest in detail

With the above list of in-and outputs we know that our function (which we will call `OLSest`) will have the following architecture:

```
function [b,bse,res,n,rss,r2] = OLSest(y,x,output)
programming commands;
...
programming commands;
end
```

We will now discuss the core of the function, the programming commands that transform the input variables

```
function [b,bse,res,n,rss,r2] = OLSest(y,x,output);
% This function performs an OLS estimation
% input:  y, vector with dependent variable
%         x, matrix with explanatory variable
%         function will automatically add a constant if the first col
%         is not a vector of ones
%         output, 1 = printed output
% output:  b, estimated parameters
%         bse, standard errors for bhat
%         res, estimated residuals
%         n, number of observations used
%         rss, residual sum of squares
%         r2, Rsquared
```

All lines beginning with a % are comment lines and you should make it a habit to describe every function at the beginning and to outline what the required inputs and the outputs are. This is extremely important to facilitate the re-use of your function. Just imagine you have written a piece of code a year ago and you want to re-use it now. You will be extremely grateful for any explanation!

```
[n,k] = size(x);
xxi = inv(x'*x);
b = xxi*x'*y;
```

These commands establish the dimensions of $\mathbf{X}$, and use formula (2) to estimate the OLS coefficients which are then stored in `b`. Note that $(\mathbf{X}'\mathbf{X})^{-1}$ is saved in `xxi` as it will be used later (in the calculation of $s_{\widehat{\beta}}$) and as inverting big matrices is computing intensive we will want to avoid having to do this twice. So saving the result and re-using it is an efficient way to use the computer's limited resources.

```
res = y - x*b;
rss = res'*res;
ssq = rss/(n-k);
s = sqrt(ssq);
bse = ssq*xxi;
bse = sqrt(diag(bse));
ym = y - mean(y);
r2 = 1 - (res'*res)/(ym'*ym);
```

The commands in this section calculate the residuals (`res`), the residual sum of squares (`rss`), the coefficient estimates standard error (`bse`) and the regression's R^2 (`r2`). Again we refer to standard econometric textbooks for the details of these calculations.

```
if output
fprintf('============================================================\n');
fprintf('===== Regression Output ====================================\n');
fprintf('Obs used = %4.0f, missing obs = %4.0f \n',n,(ninit-n));
fprintf('Rsquared = %5.4f \n',r2);
fprintf('===== Estimated Model Parameters ==========================\n');
fprintf('= Par se(Par) ==================\n');
format short;
disp([b bse]);
fprintf('===== Model Statistics =====================================\n');
fprintf(' standard error = %5.4f\n',sqrt(ssq));
fprintf('RSS = %5.4f \n',rss);
fprintf('============================================================\n');
end
```

This section of the code is only activated if the third input variable `output` is true or equal to

1. In this way the user can control whether she wants this bit printed (likely if you are only performing a single regression) or not (likely if you are estimating many regressions in some bigger procedure). This bit also contains a number of commands (like (format) and (fprintf)) which may be unknown to you at this stage, but are useful when printing results to the screen. Use the MATLAB help function for some more guidance.

The actual example `OLSest` function in the file `OLSest.m` is a somewhat expanded version of this as it also calculates t-statistics, p-values and Durbin-Watson test statistics. But all these extra stats only appear in the output (if `output` = 1). It also checks whether the input matrix `X` contains a column of constants (and if not adds one) and checks for missing observations.

5 How are functions used?

So far we have described how to write a function and we understand that it is the equivalent of a drinks machine ready to be used (receiving some inputs and handing back outputs). The question remains how to use it. The best way to use them is to save the function into a new mfile (*.m) that has the same file as the function name (here `OLSest.m`).

Having done that you can use that function (say in a MATLAB script) as demonstrated in the following code extract:

```
depvar = ...; % a vector which contains the dependent variable
expvar = ...; % a matrix that contains all explanatory variables in columns,
should include a columns of 1s for constant
[bhat,bhatse,resids,obs,resss,rsq] = OLSest(depvar,expvar,0);
disp(bhat)
```

As you can see here all input and output variables have names different to those used in the code of the function itself. Let's take the first input variable `depvar` which contains a vector with the dependent variable. In this script this vector is known as `depvar`. In this function call it is handed over to the function and there it adopts the name `y` as that is the name given to the first input variable into the function `OLSest`. Inside the function the variable `depvar` is actually unknown. Also note that the first input variable was given the value 0. By doing so we ensure that the function does not print the regression output. The first output variable is given the name `bhat` in the above script. The function `OLSest` itself actually does not know that variable. However, it did calculate the OLS parameter estimate, saved it (inside the function as `b` and then handed this value back as the first output variable. Here in the script file this is then known as `bhat` and you can continue using that value.

6 Numerical test example

If you use the data in the `OLSexample.xls` spreadsheet (column 1 as dependent variable and columns 2 to 4 as explanatory variables (don't forget to include a vector of ones as constant) you should obtain the following OLS parameter estimate:

$$\widehat{\beta} = \begin{pmatrix} 0.3983 \\ 1.0574 \\ -1.9973 \\ 0.4953 \end{pmatrix} \tag{3}$$

where the first element is the estimated constant and the remaining parameters the OLS estimates associated with the variables in columns two to four respectively.