Advanced Programming in Quantitative Economics

Introduction, structure, and advanced programming techniques

Charles S. Bos

VU University Amsterdam
Tinbergen Institute

c.s.bos@vu.nl

15 – 19 August 2011, Aarhus, Denmark

Tutorial Day 2 - Morning

10.30P Tutorial

- Addresses
- Minimal blocks

12.00 Lunch

Exercise: Addresses

Create a (or multiple small) program(s) with a `main` and a function:

1. Pass an integer to the function, `return` the square.
2. Pass an integer to the function, pass the square back through an address.
3. Pass a string, e.g. `sX= "Aargus";` to the function. Can you change only the “g” to a “h”? (Maybe first try without the function: How would you change an element of a string, directly within main?)
4. Pass the array `aX= {"Aargus", 5, <2.4, 4.6>};` to the function, change the 5 to a 7, the 4.6 to its square, and the “g” to a “h”.

Ensure you *fully* understand the address thing here... Talk to the tutor if not.

Exercise: OlsGen and Sim with functions

Target of this exercise is to set up a program for a slightly larger task. The task itself is not hard, but the idea is to do it in a structured, extensible way.

Target:

- Start with a set of regressors, X (e.g. take $X = [1\ u_1\ u_2]$ with $u_i \sim U(0,1)$), and vector of parameters β (e.g. $\beta = [1; 2; 3]$). Assume we use $n = 20$ observations for this exercise.
- Repetitively, say S times, generate n observations from $y = X\beta + \sigma\epsilon$,
- For each iteration, estimate and save for later use the parameter estimates $\hat{\beta}$ and residual standard deviation s,

$$s^2 = \frac{1}{n-k}\sum e_i^2, \qquad\qquad e = y - X\hat{\beta}$$

- After the computations, provide interesting output.

For the exercise, start e.g. with $S = 1000$, using $\sigma = 2$.

Exercise: OlsGen and Sim II

1. Analyse the exercise: What variables do I need for initial settings; what separate tasks do I have; hence, what routines could I use; what are inputs and outputs to those routines; what is the final output. *Write, on paper, an indication of the plan for your program!*
2. Start the programming, but in steps: First write `olssim0.ox`, containing only the outline of the program including the headings of the routines, then `olssim1.ox` which does the initialisation, when it works move to `olssim2.ox` which forgets about the loop, estimates just once the model and prints the outcome, then a third step in `olssim3.ox` where you add the loop and collect results (maybe start with $S = 5$?), etc.
3. Check the output: Can you print means and standard deviations of $\hat{\beta}, s$? Are those values 'expected'?

Exercise: OlsGen and Sim III

In each iteration, estimate both

$$b = \hat{\beta} = (X'X)^{-1}X'y$$

$$s^2 = \frac{1}{n-k}e'e \qquad\qquad e = y - X\hat{\beta}$$

and save `b` and `s`, e.g. in a matrix `mOLS`:

$$\texttt{mOLS} = \begin{pmatrix} b_1 & b_2 & b_3 & s \\ b_1 & b_2 & b_3 & s \\ \vdots & \vdots & \vdots & \\ b_1 & b_2 & b_3 & s \end{pmatrix} \begin{array}{c} \text{for replication } 1 \\ \text{for replication } 2 \\ \vdots \\ \text{for replication } M. \end{array}$$

Exercise: Output

Output could be in the form of text or possibly of graphs. Some exemplifying code (check manual for explanation!):

Listing 1: olsprint.ox

```
  mMS= <1, 2, 3, 4; .5, .7, .6, .8>;
  print ("%c", {"b1", "b2", "b3", "s"},       // Column names
         "%r", {"mean", "std"},               // Row names
         "%cf", {"%8.3f", "%8.3f", "%8.3f", "%8.3f"}, // Column formats
         mMS);                                  // Matrix to print
```