

Principles of Programming in Econometrics

Introduction, structure, and advanced programming techniques

Charles S. Bos

Vrije Universiteit Amsterdam
Tinbergen Institute

`c.s.bos@vu.nl`

August 2020 – Version Python

Separate lecture slides

Compilation: July 27, 2020

Overview

Principles of Programming in Econometrics

D0: Syntax, example 2⁸

D1: Structure, scope

D2: Numerics, packages

D3: Optimisation, speed

Day 1: Structure

9.30 Introduction

- ▶ Programming in theory
- ▶ Science, data, hypothesis, model, **estimation**

Structure & Blocks (Droste)

Further concepts of

- ▶ Data/Variables/Types
- ▶ Functions
- ▶ Scope, globals

13.30 Practical

- ▶ Regression: Simulate data
- ▶ Regression: Estimate model

Programming in Theory

Plan ahead

- ▶ Research question: What do I want to know?
- ▶ Data: What inputs do I have?
- ▶ Output: What kind of output do I expect/need?
- ▶ Modelling:
 - ▶ What is the structure of the problem?
 - ▶ Can I write it down in equations?
- ▶ Estimation: What procedure for estimation is needed (OLS, ML, simulated ML, GMM, nonlinear optimisation, Bayesian simulation, etc)?

Closer to practice

Blocks:

- ▶ Is the project separable into blocks, independent, or possibly dependent?
- ▶ What separate routines could I write?
- ▶ Are there any routines available, in my own old code, or from other sources?
- ▶ Can I check intermediate answers?
- ▶ How does the program flow from routine to routine?

... names:

- ▶ How can I give functions and variables names that I am sure to recognise later (i.e., also after 3 months)?
Use (always) sensible **Hungarian notation**

Even closer to practice

Define, **on paper**, for each routine/step/function:

- ▶ What inputs it has (shape, size, type, meaning), exactly
- ▶ What the outputs are (shape, size, type, meaning), also exactly...
- ▶ What the purpose is...

Also for your main program:

- ▶ Inputs can be *magic numbers*, (name of) *data file*, but also specification of model
- ▶ Outputs could be screen output, file with cleansed data, estimation results etc. etc.

Elements to consider

- ▶ Explanation: Be generous (enough)
- ▶ Initialise from main
- ▶ Then do the estimation
- ▶ ... and give results

Listing 1: stack/stackols.py

```
def main():  
    # Magic numbers  
    sData= 'data/stackloss.csv'  
    sY= 'Air Flow'  
    asX= ['Water Temperature', 'Acid Concentration', 'Stack Loss']  
  
    # Initialisation  
    ...  
  
    # Estimation  
    ...  
  
    # Output  
    ...
```

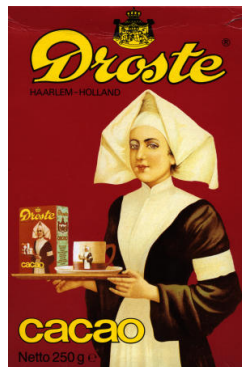
NB: These steps are usually split into separate functions

The 'Droste effect'

- ▶ The program performs a certain function
- ▶ The main function is split in three (here)
- ▶ Each subtask is again a certain function that has to be performed

Apply the Droste effect:

- ▶ Think in terms of functions
- ▶ Analyse each function to split it
- ▶ Write in smallest building blocks



Preparation of program

What do you do for preparation of a program?

1. Turn off computer
2. On paper, analyse your inputs
3. Transformations/cleaning needed? Do it in a separate program...
4. With input clear, think about output: What do you want the program to do?
5. Getting there: What steps do you recognise?
6. Algorithms
7. Available software/routines
8. Debugging options/checks

Work it all out, before starting to type...

KISS

KISS

Keep it simple, stupid

Implications:

- ▶ Simple functions, doing one thing only
- ▶ Short functions (one-two screenfuls)
- ▶ With commenting on top
- ▶ Clear variable names (but not too long either; Hungarian)
- ▶ Consistency everywhere
- ▶ Catch bugs before they catch you

See also:

- ▶ <https://www.kernel.org/doc/Documentation/process/coding-style.rst> (General Kernel)
- ▶ <https://www.python.org/dev/peps/pep-0008/> (PEP 8: Python coding guide)