

# 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 2<sup>8</sup>

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

## Target of course

- ▶ Learn
- ▶ structured
- ▶ programming
- ▶ and organisation
- ▶ (in Python/Julia/Matlab/Ox or other language)

Not: Just learn more syntax...

Remarks:

- ▶ Structure: Central to this course
- ▶ Small steps, simplifying tasks
- ▶ Hopefully resulting in: Robustness!
- ▶ Efficiency: Not of first interest... (Value of time?)
- ▶ Language: Theory is language agnostic

## What? Why?

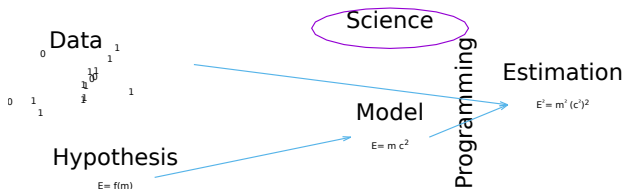
Wrong answer:

*For the fun of it*

A

correct answer

*To get to the results we need, in a fashion that is controllable, where we are free to implement the newest and greatest, and where we can be 'reasonably' sure of the answers*



## Aims and objectives

- ▶ Use computer power to enhance productivity
- ▶ Productive Econometric Research:  
combination of interactive modules and programming tools
- ▶ Data Analysis, Modelling, Reporting
- ▶ Accessible Scientific Documentation (no black box)
- ▶ Adaptable, Extendable and Maintainable (object oriented)
- ▶ Econometrics, statistics and numerical mathematics  
procedures
- ▶ Fast and reliable computation and simulation

## Options for programming

	GUI	CLI	Program	Speed	QuanEcon	Comment
EViews	+	-	-	±	+	Black box, TS
Stata	±	+	-	-	-	Less programming
Matlab	+	+	+	+	±	Expensive, other audience
Gauss	±	±	+	±	+	'Ugly' code, unstable
S+/R	±	+	+	-	±	Very common, many packages
Ox	+	±	+	+	+	Quick, links to C, ectrics
Python	+	+	+	+	±	Neat syntax, common
Julia	+	+	+	++	+	General/flexible/difficult, quick
C(++)/Fortran	-	-	+	++	-	Very quick, difficult

Here: Use Ox Matlab Python as environment, apply theory elsewhere

## History

There was once...

Apple II, CPU 6502, 1Mhz, 48kB of memory...

Now: More possibilities, also computationally:

Timings for OLS (30 observations, 4 regressors):

2017	I5-7Y54 1.2Ghz	64b	1.047.000 <sup>†</sup> /sec
2014	I5-4460S 2.9Ghz	64b	1.100.000 <sup>†</sup> /sec
2012	Xeon E5-2690 2.9Ghz	64b	950.000 <sup>†</sup> /sec
2009	Xeon X5550 2.67Ghz	64b	670.000 <sup>†</sup> /sec
2008	Xeon 2.8Ghz	OSX	392.000 <sup>†</sup> /sec
2006	AMD3500+	64b	320.000 <sup>†</sup> /sec
2004	PM-1200		147.000 <sup>†</sup> /sec
2001	PIII-1000		104.000 <sup>†</sup> /sec
2000	PIII-500		60.000/sec
1996	PPro200		30.000/sec
1993	P5-90		6.000/sec
1989	386/387		300/sec
1981	86/87 (est.)		30/sec

Increase:

$\approx \times 1000$  in 15 years

$\approx \times 10000$  in 25 years.

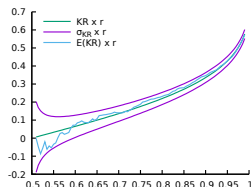
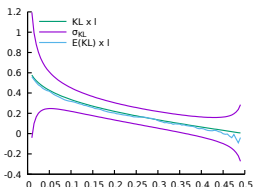
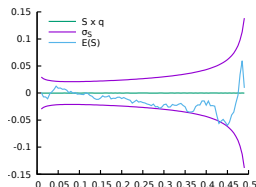
Note: For further speed increase, use multi-cpu.



## Speed increase — but keep thinking

$$x \sim \text{NIG}(\alpha, \beta, \delta, \mu) \quad P(X < x) = \int_0^x f(z) dz = F(x) \quad x_q = F^{-1}(q)$$

$$\mathcal{S}(q) = \frac{x_{1-q} + x_q - 2x_{\frac{1}{2}}}{x_{1-q} - x_q} \quad \mathcal{K}^L(q) = \frac{x_{\frac{1-q}{2}} + x_{\frac{q}{2}} - 2x_{\frac{1}{4}}}{x_{\frac{1-q}{2}} - x_{\frac{q}{2}}} \quad \mathcal{K}^R(q) = \dots$$

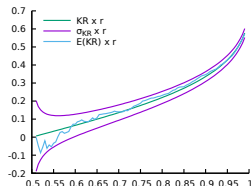
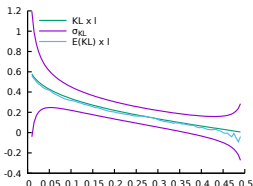
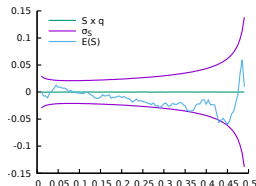


Direct calculation of graph: > 40 min

## Speed increase — but keep thinking

$$x \sim \text{NIG}(\alpha, \beta, \delta, \mu) \quad P(X < x) = \int_0^x f(z) dz = F(x) \quad x_q = F^{-1}(q)$$

$$\mathcal{S}(q) = \frac{x_{1-q} + x_q - 2x_{\frac{1}{2}}}{x_{1-q} - x_q} \quad \mathcal{K}^L(q) = \frac{x_{\frac{1-q}{2}} + x_{\frac{q}{2}} - 2x_{\frac{1}{4}}}{x_{\frac{1-q}{2}} - x_{\frac{q}{2}}} \quad \mathcal{K}^R(q) = \dots$$



Direct calculation of graph: > 40 min  
Pre-calc quantiles (=memoization): 5 sec