

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: August 3, 2020

Overview

Principles of Programming in Econometrics

D0: Syntax, example 2⁸

D1: Structure, scope

D2: Numerics, packages

D3: Optimisation, speed

Day 2: Numerics and flow

9.30 Numbers and representation

- ▶ Steps, flow and structure
- ▶ Floating point numbers
- ▶ Practical Do's and Don'ts
- ▶ Packages
- ▶ Graphics

13.30 Practical

- ▶ Cleaning OLS program
- ▶ Loops
- ▶ Bootstrap OLS estimation
- ▶ Handling data: Inflation

Reprise: What? Why?

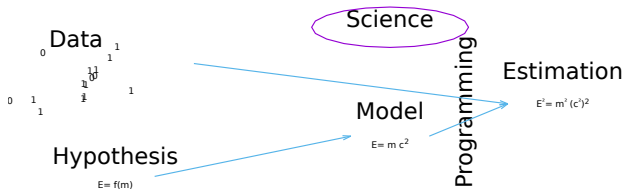
Wrong answer:

For the fun of it

A

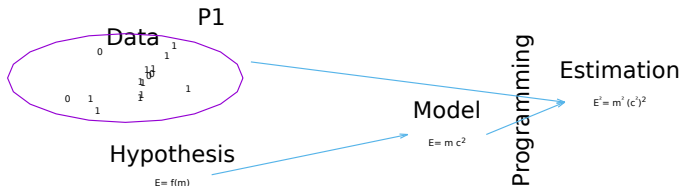
correct answer

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



Step P1: Analyse the data

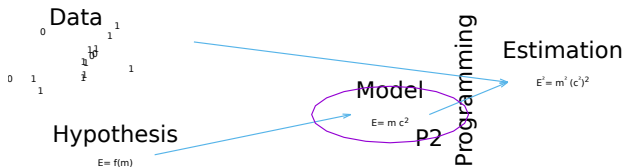
- ▶ Read the original data file
- ▶ Make a first set of plots, [look at it](#)
- ▶ Transform as necessary (aggregate, logs, first differences, combine with other data sets)
- ▶ Calculate statistics
- ▶ Save a file in a convenient format for later analysis



```
mData= np.hstack([vDate, mFX])
df= pd.DataFrame(mData, columns=["Date", "UKUS", "EUUS", "JPUS"])
df.to_csv("data/fx9709.csv")
df.to_csv("data/fx9709.csv.gz", compression="gzip")
df.to_excel("data/fx9709.xlsx")
```

Step P2: Analyse the model

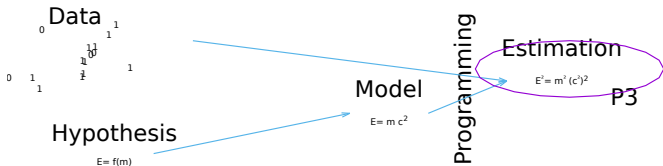
- ▶ Can you simulate data from the model?
- ▶ Does it look 'similar' to empirical data?
- ▶ Is it 'the same' type of input?



```
mU= np.random.randn(iT, 4);    # Log-returns US, UK, EU, JP factors
mF= np.cumsum(mU, axis=0);      # Log-factors
mFX= np.exp(mF[:,1:]-mF[:,0]);  # FX UK EU JP wrt US
```

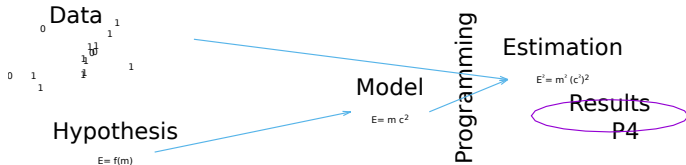
Step P3: Estimate the model

- ▶ Take input (either simulated or empirical data)
- ▶ Implement model estimation
- ▶ Prepare useful outcome



Step P4: Extract results

- Use estimated model parameters
- Calculate policy outcome etc.



Step P5: Output

- ▶ Create tables/graphs
- ▶ Provide relevant output

Often this is the hardest part: What exactly did you want to know? How can you look at the results? How can you go back to original question, is this really the (correct) answer?

Result of steps

```
def main():  
    # Magic numbers  
    sData= "data/fx0017.csv"           # Or use "data/sim0017.csv"  
    asFX= ["EUR/USD", "GBP/USD", "JPY/USD"]  
    vYY= [2000, 2015]                 # Years to analyse  
  
    # Initialise  
    (vDate, mRet)= ReadFX(asFX, vYY, sData)  
  
    # Estimate  
    (vP, vS, dLnPdf)= Estimate(mRet, asFX)  
    mFilt= ExtractResults(vP, mRet)  
  
    #Output  
    Output(vP, vS, dLnPdf, mFilt, asFX)
```

- ▶ Short main
- ▶ Starts off with setting items that might be changed: Only up front in main (*magic numbers*)
- ▶ Debug one part at a time (t.py)!
- ▶ Easy for later re-use, if you write clean small blocks of code
- ▶ Input for estimation is *prepared* data file, not raw data (...).