Agent-Based Modeling & Object Oriented Programing

Schedule

Time	Sat	Sun	Mon	Tue	Wed	Thu
9.00-10.30	Intro & Organization	Scale-free distributions	Dynamical systems	Lab 4 - ABM	Distributions & entropy	Wrap-up
11.00-12.30	Meta-theoret. foundations & history	Network theory	Lab 3 - Dynamical systems	Lab 4 - ABM	Lab 5 - Distributions & ABM	Wrap-up
12.30-14.30			Lunch break			
14.30-16.00	Lab 1 - Foundations	Lab 2 - Networks	OOP & ABM		Lab 5 - Distributions & ABM	

Claudius
Torsten
Claudius & Torsten
Lab session
Wrap-up

Motivation

- The story so far:
  - Economies are ...

  "... a complex object, every part or component of which is connected with other parts of the same object in such a manner that the whole possesses some features that its components lack — that is, emergent properties" (Bunge, 1990).

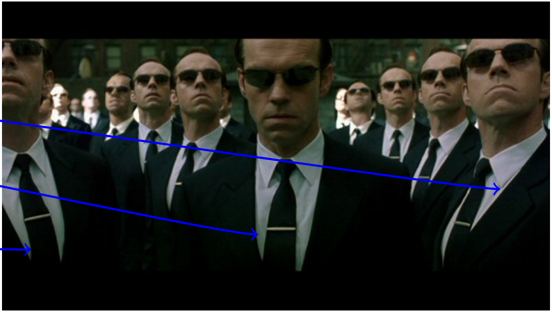
- Network theory as a language to describe relation
- Dynamics as a language to describe dynamics
- Game theory as a language to describe interactions
- Bringing all together is difficult: requires exact, but flexible language...
  - ...an algorithmic language as 'the right mathematics for the social sciences'

Agent-Based Modeling

- In social sciences (including economics) one of the main purposes of simulation is to investigate situations with many interacting agents (people, firms, governments, ...)
- The resulting simulations construct large numbers of similar "agents" (program objects) that differ only in variable values or minor properties
- The program proceeds to let these agents interact.
- Though not generally different from other simulations this approach has come to be called *agent-based simulation*.

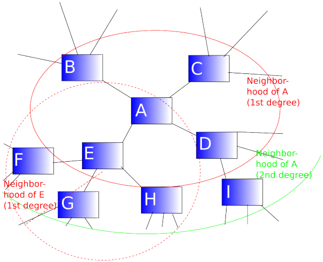
Agent-Based Modeling

- x_1
  y_1
- x_2
  y_2
- x_3
  y_3
- ...



Agent-Based Modeling

- Agent-based modeling (with or without simulation) in general allows direct modeling of
    - heterogenity between elements of economic systems ("agents")
    - (heterogenous) direct interactions
    - networks between agents
- ABM thus also allows to investigate the influence of heterogeneity etc. on macro-level outcomes of interactions.
- Due to the complexity of such models, it is convenient to employ simulation as a tool of analysis.



Agent Based Modelling

- Formally: a system of difference equations
  - Computers must always discretize
- Definition according to Tesfatsion (2017): *locally-constructive sequential games*
  - Local: interaction between parts is directly modeled
  - Constructive: the model *generates* its output
  - Sequential: agents do things one after another
  - Games: interaction of interested elements takes place
- Helps to avoid assumptions made only for the sake of analytical tractability
- Builds upon 7 modeling principle

ABM and the Lucas Critique

- Reference point for DSGE and CGE:
  - Stagflation in the 70s
  - Misleading policy advice based on the Cowles Commission approach
- Lucas critique (1976): necessity for sound micro foundations of economic analysis and consideration of expectations
  - An epistemological change w.r.t. to what counts as an 'explanation'

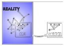
ABM and the Lucas Critique

- Methodological consequence: Use of rational expectations (RE)
  - Focus on RE reduces the degrees of freedom of the complexity reduction function
  - No real micro calibration of assessment of intermediate results, e.g. Euler equation
- Features such as true uncertainty and the corresponding heuristics (Gigerenzer et al.) are excluded from analysis
- ABM as the ‚better‘ answer to the Lucas critique?!

ABM, OOP, and Systemism

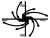
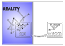
- To specify agents in an ABM we often exploit the OOP features of Python
- Agents should be defined as a class
- Classes as 'blueprint' to create a large population of agent without difficulties
- Thus, you may also have a class `model`
- There is a deeper meta-theoretical connection between 'systemism' and OOP
    - $\langle \text{Systems, Mechanisms} \rangle \leftrightarrow \langle \text{Objects, Methods} \rangle$

ABM: Workflow

1. Define SUI 

2. Algorithmize SUI 
3. Program simulation 
4. Define initial variables

$$s(0) = s_0$$
$$x(0) = x_0$$

5. Run simulation 
6. Validate simulation 
7. Verify simulation results 
8. Draw conclusions (for the SUI) 

ABM: Randomness

- Most ABM involve randomness
- Usually, two runs of the ABM do not yield the exactly same results
- To study the model, you need to run it many times and analyse it statistically
  - There is literature and techniques inspired by experimental research to answer questions such as “How many runs are needed”
- Python offers many random number generators, particularly through the numpy

The Epistemological Benefits

- ABM are open to all kind of model validation...
  1. Input validation
  2. Process validation
  3. Descriptive and predictive validation
- ABM are flexible and may include many mechanisms
- ABM can be related to analytical models
  - Helps to highlight and defend pluralism...
  - ...but also to make it work
    - Relate different research programs to each other
- *Policy readiness*: bridge between academics, and applied large-scale models

Main Challenges

- ABM are comparatively hard to verify
- Different to communicate, no unique theoretical core
- More freedom on the assumption level necessarily introduces more arbitrary parameters

Calibration

- Recall that there are different approaches to simulation: KISS (keep it simple, mechanism-oriented), KIDS (keep it descriptive, data-oriented).
- Calibration vs. validation

Θ - Parameters
y - Calibration data

F - Fitness function
x - Simulation output

$$\Theta^* = \arg\max F(y, x(\Theta))$$

Calibration: Finding this

Validation: Choosing that

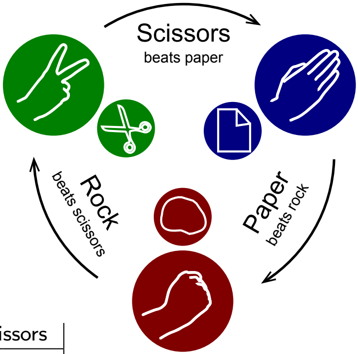
- Calibration techniques
  - Kriging (Gaussian process interpolation)
  - Genetic algorithms (differential evolution etc.)
  - Sampling techniques: Simple ones: parameter grid, sophisticated ones: NOLH (nearly orthogonal lating hypercube) sampling.

Some Suggestions for Model Building

- Start with writing comments of what you want to do first
- Do not start with writing classes immediately
  - Test their elements outside a class
- Write a simple interaction, once it works make a look
- Once the '`for i in t` '-loop works, use functions
- As soon as the loop works, build a simulation class '`model`'
  - In the init function, this class should create everything that is needed for the simulation
  - It should feature the following functions:
    - a function '`go`' that goes through one particular time step
    - a function '`run`' that runs the mode for t timesteps
    - a function '`save`' that saves/analyzes the results
  - Sometimes it is also useful to write a meta-function '`main`' that calls '`model`' often, stores all results and later calculates summary statistics

Agent-Based Simulation: Example

- A population of agents interacts in a way that is conveniently modeled as rock-paper-scissors games with random matching.
- The population is subject to evolution with fitnesses proportional to average payoffs.



		Player 2		
		Rock	Paper	Scissors
Player 1	Rock	b b	c a	a c
	Paper	a c	b b	c a
	Scissors	c a	a c	b b

$c > b > a$