Python II module

Lecturer: Pieter Kleer

Computational Aspects in Econometrics (35V3A1-B-6), 2023-2024

Tilburg University

Understanding Society

About me

- Assistant professor in Department of Econometrics and **OR**
  - Since April 2021.

I'm the **coordinator** of this course:

- Contact me with general questions about course, exam, etc…
- Module-specific questions always go to lecturer of module.

Matlab vs. Python

Similar functionalities

- Matlab (and its toolboxes)
- Python combined with a selection additional packages
  - Such as NumPy, SciPy, Matplotlib, and Scikit-learn.

Advantages Python

- Free
- Popular
- Fast development in data science and machine learning

Matlab will be covered in last module of this course.

Schedule Python II module

Lecture 1-2: NumPy

- The basics to do scientific computation using arrays (vectors/matrices).

Lecture 3-4: SciPy

- Root finding, optimization, statistics, curve fitting.

Lecture 5: Scikit-learn

- Some machine learning algorithms for classification problems.
- Logistic regression, *k*-Means algorithm, *k*-Nearest Neighbour (kNN) algorithm.

Course materials

based mostly on materials of Ruud Brekelmans (including this presentation).

Software

- *Recommended:* Anaconda installation
- Includes Python (Spyder), Jupyter, JupyterLab and more…

Weekly lecture material

- Lecture materials: https://pskleer2.github.io
- Solutions to exercises can be found there as well.

Assignment

- Will be asked to implement certain algorithmic tasks.
  - Assessment: Correctness and “Good coding” (next slide).
- Same groups as for Emiel’s assignments.
- You have to write report (couple of pages) as well.
  - High-level overview of what you implemented.
  - *“Who did what?”* section.

Good coding: What's important?

- **Efficient computations**: Use NumPy ('vectorize' operations whenever possible)
- **No hard coding**: Replace problem data by variables.
- **DRY** (Don't Repeat Yourself): Make a function/loop for things that you repeat.
- **Single responsibility**: Split larger problem into subproblems (functions).
- **Coding style & documentation**: Try to follow general Python coding practices.
  - Function documentation between triple double-quote characters starting with one main header line.
  - Clearly describe what a function does and what its input and output arguments are.
  - Choose descriptive variable names, lines not longer than 80 characters.
  - Don't add comments for every line. Add comments for main ideas and complex parts.
    - A comment should not repeat the code as text (e.g. "time = time + 1 # increase time by one).
- **KISS** (Keep It Simple Stupid): Simplicity above complexity.

See lec1-goodcoding-examples.py (or Notebook) for illustrations of these concepts.