Programming Practices for Research in Economics

Christian Zünd, Dora Simon, Ursina Schaede, Ulrich Bergmann, Carlo Zanella

Motivation

Much of modern economics research involves the researcher spending their lives in front of a computer – analysing data or simulating economic models.

Unfortunately, it is rare that they have been taught how to do this well. Class exposure to programming languages is most often limited to the simple use of Stata and Matlab to solve ‘toy’ examples designed to illustrate a theoretical result or implement a method with known properties and ex-ante known results. These skills do not scale up in a straightforward manner to handle complex projects that make up research papers, PhD theses or typical work in government or private business settings. As a result, young economic researchers spend too much time wrestling with software and too little time doing economics – where their advantage generally lies. Even with their large investments in software wrestling, we typically have no idea how reliable or efficient their programs are.

This course is designed to sort out this messy state of affairs. It is aimed at PhD students who expect to write their theses in a field that requires modest to heavy use of computation. Examples include applied microeconomics, econometrics, macroeconomics, computational economics and any field that either involves real-world data or does not generally lead to models with simple closed-form solutions. We will introduce students to programming methods that will substantially reduce their time spent programming while at the same time making their programs more dependable and their results reproducible.

The course draws extensively on some simple techniques that are the backbone of modern software development which most economists are simply not aware of. It shows the usefulness of these techniques by means of hands-on examples for a wide variety of economic and econometric applications.

Target Audience

This course is intended for PhD students who are transitioning from coursework to research. Next to your economics background, we will only assume that you have written small pieces of code before, like Stata .do-files or Matlab .m-files for problem sets in your Masters degree or first-year PhD classes. Knowledge of a specific programming language is not required.

A large part of this course is really about tool choice. We will take care in pointing out which language is most appropriate for which problem, and provide you with introductions to three popular choices for data- and computationally intensive computing. These are not the only choices available but some knowledge of these languages will make picking up others on your own or through our PhD computing seminars relatively easy.

Course Objectives

This course has two closely intertwined objectives:

1. Enhancing students' programming efficiency.
2. Providing the tools to make data analysis and computation reproducible.

Learning objectives for specific modules will be provided within the Course Notes.

Evaluation

The course is evaluated on a pass/fail basis. There will be a final assignment that is due four weeks after the course concludes. This assignment will count 100%. More information will be provided before the course begins.

Rules of the Game

The class is designed to be 'hands-on' in the sense that you will be programming a lot of things *during the class.* We strongly believe the only way to learn programming is to do programming. Please bring your laptop with you to each session and install the required software before the course begins. Try to complete each activity we do in class and be prepared to ask and answer questions during class. Slides or notes will be made available at the beginning of each day, codes that solve exercises will be posted during or after the session.

Office Hours

Due to the intensive nature of the course, we have decided not to schedule office hours. Feel free to talk to us before and after each session throughout the course and ask many questions during each session.

Times and Locations

- Dates: Daily from August 27th until September 14th (excluding weekends)
- Morning Session: TBA
- Afternoon Session: TBA
- Location: TBA

Preliminary Programme

The following is a preliminary programme. It may be updated prior to the beginning of the course, and updated schedule will be forwarded before the course begins.

	Monday	Tuesday	Wednesday	Thursday	Friday
Week 1					
Morning	Introduction	Version Control	Python Basic	Python Pandas	Python Project
Afternoon	Terminal	Python Basic	Python Numpy	Python Project	Python Project
Week 2					
Morning	Webscraping	Webscraping	QGIS	R	R
Afternoon	Webscraping	QGIS	R	R	R
Week 3					
Morning	R	Automatization	Machine Learning	Machine Learning	ML Project
Afternoon	R	Automatization	Machine Learning	Machine Learning	ML Project

Topics in Depth

Terminal and Version Control

You will learn how to use the terminal, the nerdy black window with the white font, on your computer. It's nerdy, but useful. Version control helps to avoid having five times the same document with the ending v1, v2, v3, FINAL, FINAL-FINAL.

Python

Python is a real programming language that is used by real programmers. Numpy and Pandas are so-called libraries that store commands you will find useful for your research.

Webscraping and QGIS

Webscraping enables you to extract information from websites and store it in a data file. QGIS is used when working with geospatial data (e.g. maps). Both of these tools are used a lot and you will very likely encounter them at some point of your research career.

R

R is a programming language dedicated to data analysis. It is great for data manipulation, regressions and graphs - everything a researcher cares about.

Automatization

You will learn how you can assemble all your little codes into a big research project in the most efficient way. Also, you will learn how to do the same procedure several times without copy-pasting code.

Machine Learning

Standard programming gives your computer specific instructions on how to fulfill a task. Machine Learning tells your computer to achieve a certain goal without specifying every little step but letting it find out how to achieve the goal in the best way.