Programming Practices for Research in Economics

Ulrich Bergmann Matteo Courthoud Lachlan Deer

Winter 2020

E-mail: pp4rs.contact@gmail.com
Web: pp4rs.github.io/2020-uzh
Dates: 27th January - 14th February
Class Hours: 9:30am-12:30pm, 2pm - 5pm
Class Days: daily, Monday - Friday
Class Room: To Be Announced

Motivation

Much of a researchers live in modern economic research is spent in front of a computer – be it to analyze data or to simulate economic models.

Unfortunately, it is rare that we have been taught how to do this in a structured and efficient way. 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, like copy and pasting pieces of code repeatedly, 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 – our comparative advantage. Even with large investments in software wrestling and self-learned skills, we typically have no idea how reliable or efficient our programs are.

This course is designed to improve learners' programming abilities. 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.

The course introduces students to a set of industry standard tools and programming methods that aim to reduce time spent programming while at the same time making programs more dependable and results reproducible. It draws extensively on basic techniques and tools that are the backbone of modern software development and have enabled the development and maintainance of Windows or the Google search engine. At the same time, these tools are simple enough to be used in small research projects, too. The course shows the usefulness of these techniques and trains the student in them 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 two popular choices for

data- and computationally intensive computing. We also introduce a tool kit designed to improve the replicability of your code. The programming languages and tools introduced in the course are not the only choices available but some knowledge of these languages and best practices will make picking up others on your own relatively easy by providing a solid basic training.

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 to not 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 27th January until 14th February (excluding weekends)
- Morning Session: 9.30 - 12.30
- Afternoon Session: 14.00 - 17.00
- 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:					
AM	Terminal*	Basic Python	Basic Python	Python: Pandas	Python: Metrics
PM	Terminal*	Basic Python	Python: Numpy	Python: Plotting	Python: SciPy
Week 2:					
AM	Webscraping	Python Project	Version Control*	Version Control*	R: Basics*
PM	Adv. Python	Python Project	Version Control*	R: Basics*	R: Data Analy.*
Week 3:					
AM	R: Plotting*	R: Econometrics	R Project	Build Tools*	Build Tools
PM	R: Econometrics*	Advanced R	R Project	Build Tools*	Build Tools

Students are expected to have completed the Installation Guide and successfully installed all required software for the course prior to the first day.

1st Year Students vs. 2nd Year Students

In this edition we have two distinct groups of students:

- 1st year PhD Students, who have not taken any existing PP4RS courses
  - We expect you to attend **all sessions**
- 2nd year PhD Students, who have taken the PP4RS: Foundations course in 2019
  - Can skip sessions marked with a *, but can attend as a refresher
  - We expect you to be fluent in these topics

The exact topics covered in each session market with a * might differ slightly from the 2019 version as we have more time and our thoughts on what is important to cover may have evolved over time.

Brief Topic Outlines

Terminal

The Unix shell has been around longer than most of its users have been alive. It has survived so long because it's a power tool that allows people to do complex things with just a few keystrokes. More importantly, it helps to combine existing programs in new ways and automate repetitive tasks to avoid typing the same things over and over again. Use of the shell is fundamental to using a wide range of other powerful tools and computing resources (including "high-performance computing" and cloud computing resources). These lessons will start you on a path towards using these resources effectively.

Python Programming Language

Python is one of the most popular and fastest growing programming language because of its ease of use and power. It has also become the most used statistical software in the recent years. These modules have two distinct goals: (1) introduce basic programming syntax, and (2) introducing specific packages that are useful for computational analysis and data-driven research. We introduce basic programming syntax using the Python language because of its simplicity to get started for novice users. The introductions to specific packages are designed to highlight how to solve problems that are typically encountered by economics and business researchers – such as solving models using computational

techniques, the automated collection of data from websites or applied microeconometric modelling, using the Python language. We emphasize best practice technqiues as we progress through the material.

Version Control

Version control is the lab notebook of the digital world: it's what professionals use to keep track of what they've done and to collaborate with other people. Every large software development project relies on it, and programmers use it for their small jobs as well. And it isn't just for software: books, papers, small data sets, and anything that changes over time or needs to be shared can and should be stored in a version control system. Teams are not the only ones to benefit from version control: lone researchers can benefit immensely. Keeping a record of what was changed, when, and why is extremely useful for all researchers if they ever need to come back to the project later on (e.g., a year later, when memory has faded or a former version of a file has been overwritten).

The R Programming Language

The goal of this lesson is to teach novice programmers to write modular code and best practices for using R for data analysis. R is free, computationally fast, and has a wide array of third-party packages for almost all imaginable statistical applications. It is the second most used statistical software today and the most popular one among academic researchers. The emphasis of these materials is to give attendees a strong foundation in the fundamentals of R and to teach best practices for scientific computing: breaking down analyses into modular units, task automation, and encapsulation. This workshop will focus on teaching the fundamentals of the programming language R, data wrangling, data visualization and regression techniques common to applied microeconomics researchers.

Build Tools

Build Tools can run commands to read files, process these files in some way, and write out the processed files. For example, we can:

- Run analysis scripts on raw data files to get data files that summarize the raw data;
- Run visualization scripts on data files to produce plots and statistical tables; and to
- Parse and combine text files, tables and plots to create papers.

Most importantly, Build Tools track the dependencies between the files they create and the files used to create them. This allows build tools to only recompute necessary steps after a data or code file has been changed – without the need to run the whole project from the beginning.