Programming Practices for Research in Economics: Foundations

Lachlan Deer Julian Langer

Winter 2019

E-mail: pp4rs.contact@gmail.com
Web: pp4rs.github.io/2019-foundations-uzh
Dates: 13 Feb 2019 to 16 Feb 2019
Class Hours: 9:30am-12:30pm, 2pm - 5pm
Class Days: daily, Wednesday to Saturday
Class Room: To Be Announced

Motivation

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

Exposure to programming languages in the current graduate curricula 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, graduate researchers in economics and business spend too much time wrestling with the software required to produce their research.

This course is designed to assist graduate and early career researchers become more fluent in conducting computational research. 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. We introduce students to software and 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 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 about acquiring skills to enhance the reproduciblity of results generated from computational processes. We will take care in pointing out the current challenges

in acheiving reproducible research workflows, and provide you with introductions to popular choices for data- and computationally intensive computing. The tools introduced are not the only choices available but are well suited to the workflow of typical research projects in economics and business schools. Knowledge of the tools introduced in this course will make picking up others on your own 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 February 13th until February 16th (Wednesday to Saturday)
- 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.

	Day 1	Day 2	Day 3	Day 4
Morning	Terminal	Version Control	R: Data Explor.	Build Tools
Afternoon	Version Control	R: Basics	R: Econometrics	Build Tools

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 them combine existing programs in new ways and automate repetitive tasks so they aren't 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" supercomputers). These lessons will start you on a path towards using these resources effectively.

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 most 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).

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 commonly used in many scientific disciplines for statistical analysis and its array of third-party packages. 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, 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 and plots to create papers.

Build Tools track the dependencies between the files they create and the files used to create these. If one of the original files (e.g. a data file) is changed, then our Build Tool software knows to recreate, or update, the files that depend upon this file (e.g. a plot).