Effective Programming Practices for Economists

1. Introduction

Hans-Martin von Gaudecker

Department of Economics, Universität Bonn

universitätbonn

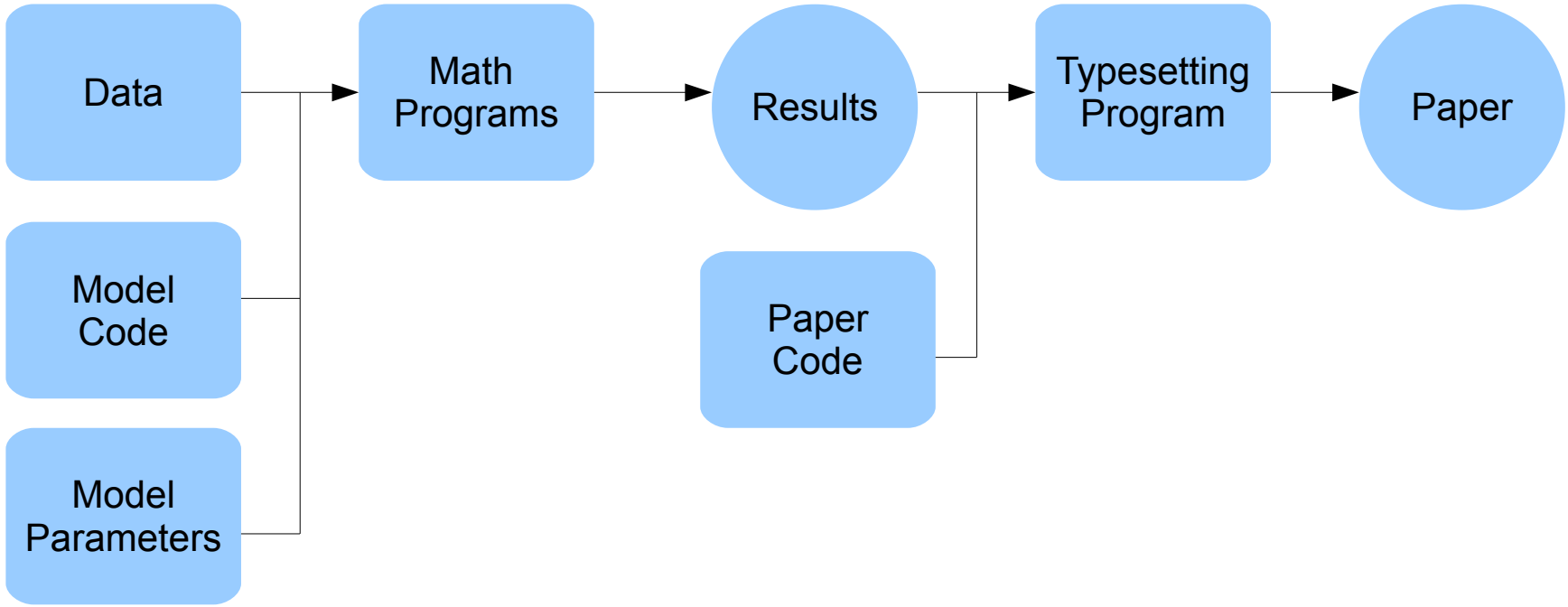
Introducing ourselves and the big picture

Course goals and how to reach them

Goals for this course[1] Novelty, Efficiency, and Trust

- Improve computing skills, so you can do things you could not do before
- Also show how can do a given set of things with less effort
- Increase the confidence in results that are produced this way (both yours and others' results)

[1] Shamelessly borrowed from `http://software-carpentry.org/blog/2013/01/novelty-efficiency-and-trust.html`


Data
Model Code
Model Parameters
Math Programs
Results
Paper Code
Typesetting Program
Paper

How to get there?
Practice, practice, practice!

Term paper

- You are completely free to pick a subject
- Could become (a chapter of) your thesis
- I do not care much about the economic content, but about the techniques you apply
  - How much does it take to understand your code?
  - Can I reproduce your results?
- We could look for something together, but tell me early
  - Before Christmas
- Due 31 March 2017

A bird's eye view of the course contents

Programming?

- “Traditional” computational economics / econometrics
  - Efficient algorithms for solving models
  - Intractable / too much data
- Here:
  - Tools for managing the research workflow
  - Entails some basic programming techniques as well
  - “Software engineering” (**software carpentry**)
- Whatever our precise application – it is **hard**

Core topics (reproducibility and quality assurance)

5, 7, 11 Keeping track of versions and progress, collaboration

12 The magic button

9 Debugging

13 Documentation

15 Testing

And the rest?

2, 3, 4 Some math, the shell, and LaTeX

6, 8, 10 Very basic programming, some scientific tools

14 Object-orientation

16, 17 Handling text and data

18 Overviews on project layouts and tool choice

Some thoughts about scientific progress, the publishing process, and verifiability / reproducibility

Scientific progress requires reproducibility

- Standing on the shoulders of giants . . .
- Works because scientific results are routinely verifiable
- Verification necessary because errors creep in everywhere

Scientific progress requires reproducibility

- Economic theory:
  - Mathematisation response to errors in verbal reasoning
- Mathematics:
  - Requiring proofs response to errors in mathematical reasoning

Scientific progress requires reproducibility

- Empirical economics:
  - Hypothesis testing response to seeing patterns in noise
  - Requiring descriptions of populations of interest and sampling schemes, rules for identification of causal effects, etc., response to fact that every detail of the analysis may be relevant for the conclusions.
- Computations in {empirical, macro, finance, . . . } papers?

  → Take a step back and think about the publishing process

The publishing process

- Results are only publishable if they are believed to be correct and reproducible
  - In practice, rely on expectations and cultural norms
  - Only actually check work that is already under suspicion
- Correctness of code rarely questioned
  - We all know programs are buggy…
  - … but did you ever hear of a paper being rejected because of concerns over software quality?

The publishing process

- Reproducibility often nonexistent
  - How many people can reproduce, much less trace, each computational result in their thesis?
- McCullough and Vinod (2003):

> [Then this is not science and] *cannot be trusted either as part of the profession's accumulated body of knowledge or as a basis for policy.*

How did we get here?

- Publishing process still works like in the 1970s
  - Only slower – Ellison (2002)
  - And tougher up there these days – Card and DellaVigna (2013)
- Not well suited for reviewing results that are not verifiable by human reader alone
  - Large datasets
  - Complicated computer programs

This is not made up of thin air. . .

- Hoxby (2000) – Rothstein (2007a) – Hoxby (2007) – Rothstein (2007b)
  - 44 larger streams in a MSA?
- Levitt (1997) – McCrary (2002) – Levitt (2002)
  - Flipped weights
- Oreopoulos (2006, 2008)
  - Mistakes in data generation
- McCullough and Vinod (2003)
  - Comprehensive replication study

This is not made up of thin air. . .

- Original authors claim that results still go through
  - Statistical significance mostly lost.
  - Papers probably would not have been published (so well) if revised results had been there in the first place
  - Many still cited widely
- Likely the tip of the iceberg
  - Incentives for replication smaller in lower-ranked journals

Replicate or reproduce?

- Replication
  - Running the same code on the same machine, do I get the same result?
  - Minimal requirement for qualifying as science, but not satisfied by many papers
  - Replicating a finding doesn't add to knowledge base
  - Sligthly increases trustworthiness of original findings unless it discredits them

Replicate or reproduce?

- “Scientific” reproducibility
  - Independent verification of results
  - How do results depend on assumptions, data, tools, . . .
  - Adds to knowledge base
- Trade-off:
  - Writing new code to solve the same problem adds more to the knowledge base . . .
  - . . . but the incentives to do so are essentially zero

Consequences for computing-based research

Koenker and Zeileis (2009)

The transition of econometrics from a handicraft industry [. . .] to the modern sweatshop of globally interconnected computers has been a boon to productivity and innovation, but sometimes seems to be a curse. Who among us expected to be in the 'software development' business? And yet many of us find ourselves precisely in this position, and those who are not probably should be. [. . .] software development is no longer something that should be left to specialized commercial developers, but instead should be an integral part of the artisanal econometric research process. Effective communication of research depends crucially on documentation and distribution of related software and data.

Jon Claerbout

Geophysicist in Stanford and an early proponent of reproducible research:

> *An article about computational science in a scientific publication is* ***not*** *the scholarship itself, it is merely* ***advertising*** *of the scholarship. The actual scholarship is the complete software development environment and the complete set of instructions which generated the figures.*

(quoted from Buckheit and Donoho, 1995)

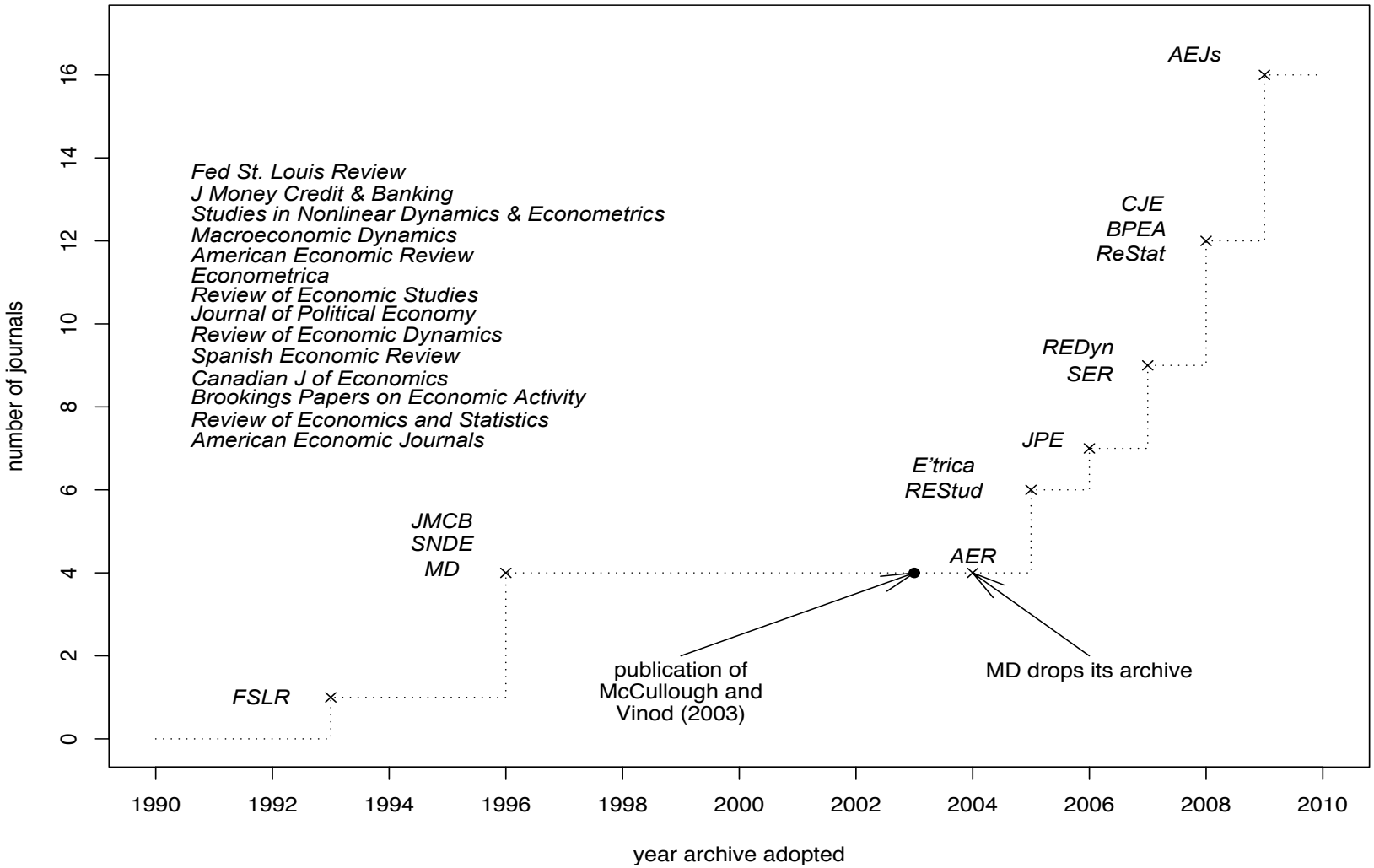
The times are a-changin'

- No stable equilibrium: Expectations will move up from zero
- Journals slowly but surely take on replication policies
- But 1970s concepts live on. From the AER website:

  Authors must provide a Readme PDF file listing all included files and documenting the purpose and format of each file provided . . .

- Other fields are faster, but also slow

Journals with mandatory data & code archives (Figure 1 in McCullough, 2009)



Toolbox for the course

Main tool: Python

- Free
  - Do not overvalue this point – opportunity cost of time
  - **But:** Very mature, stable, documented, easy to get help
- Open source
  - “Just trust” {Stata, Matlab, EViews, Gauss, . . . } functions?
  - Baiocchi (2007)

Main tool: Python

- Huge range of applicability
  - Google writes a lot of code in Python, employed its creator until he moved to Dropbox
  - Has a very large scientific community
  - Economics and econometrics communities are growing fast
- Runs smoothly on all operating systems

Main tool: Python

- Great scripting language
  - Automate boring, tedious tasks like renaming files, cleaning data, . . .
  - Sweigart (2015)
- Works well with almost anything else
  - Call R functions for advanced statistical methods
  - Write performance-critical bits in Numba or Cython (subset of Python syntax) . . .
  - Or interface directly with C or Fortran libraries

Main tool: Python

- Easy to learn
- Has a very clear syntax
- Is extremely well documented
- High-level, interpreted, dynamically typed language
  - High development speed
- Lots of tools for documenting your own stuff
  - Also things written in other languages

Three ways we'll use Python

- Exemplatory
  - Techniques you can use in any language
- The magic red button
  - Use Waf to run scientific code (Matlab m-files, Stata do-files, . . .) and to compile LaTeX documents
  - I.e., use it as a glue – Langtangen (2008)
- Documentation
  - Use Sphinx to document projects

Required programs

- See installation guide:

  `http://hmgaudecker.github.io/econ-python-environment`

- Anaconda Python 3 distribution

- Git (Command line)

Required programs (for now)

- TeXLive / MacTeXor MikTeX
- A text editor to use with it
  - Shop a bit around. Notepad not an option. Word less so
  - Personal favourite: Sublime Text (well-spent EUR 55, may try for free)
  - Recent, modern open source stuff: Light Table, Atom
  - Syntax highlighting is the most important feature
  - Emacs/Vim most comprehensive, steep learning curve
- Learn one, learn it **very** well

The Economics Collaboration Server

Preparation for Friday

- Never used the Shell (Terminal, Command Prompt) ?

  `https://www.youtube.com/watch?v=U3iNcBtycaQ&html5=True`
  `http://swcarpentry.github.io/shell-novice/`

  - Introduction
  - Files and Directories

- Math refresher
  - Skim through: `http://www.jfsowa.com/logic/math.htm`
  - Sections 1, 2, 4, 8
  - Review / take questions on Friday

References I

- Baiocchi, Giovanni (2007). “Reproducible Research in Computational Economics: Guidelines, Integrated Approaches, and Open Source Software”. In: *Computational Economics* 30.1, pp. 19–40.
- Buckheit, Jonathan B. and David L. Donoho (1995). “Wavelab and Reproducible Research”. In: *Wavelets and Statistics*. Ed. by Anestis Antoniadis and Georges Oppenheim. Lecture Notes in Statistics. Springer, pp. 55–82.
- Card, David and Stefano DellaVigna (2013). “Nine Facts about Top Journals in Economics”. In: *Journal of Economic Literature* 51.1, pp. 144–161.
- Ellison, Glenn (2002). “The Slowdown of the Economics Publishing Process”. In: *Journal of Political Economy* 110.5, pp. 947–993.
- Hoxby, Caroline M. (2000). “Does Competition among Public Schools Benefit Students and Taxpayers?” In: *American Economic Review* 90.5, pp. 1209–1238.
- – (2007). “Does Competition among Public Schools Benefit Students and Taxpayers? Reply”. In: *American Economic Review* 97.5, pp. 2038–2055.
- Koenker, Roger and Achim Zeileis (2009). “On Reproducible Econometric Research”. In: *Journal of Applied Econometrics* 24.5, pp. 833–847.

References II


Langtangen, Hans Petter (2008). *Python Scripting for Computational Science*. 3rd ed. Berlin and Heidelberg, Germany: Springer.


Levitt, Steven D. (1997). "Using Electoral Cycles in Police Hiring to Estimate the Effect of Police on Crime". In: *American Economic Review* 87.3, pp. 270–290.


– (2002). "Using Electoral Cycles in Police Hiring to Estimate the Effects of Police on Crime: Reply". In: *American Economic Review* 92.4, pp. 1244–1250.


McCrary, Justin (2002). "Using Electoral Cycles in Police Hiring to Estimate the Effect of Police on Crime: Comment". In: *American Economic Review* 92.4, pp. 1236–1243.


McCullough, B. D. (2009). "Open Access Economics Journals and the Market for Reproducible Economic Research". In: *Economic Analysis & Policy* 39.1, pp. 117–126.


McCullough, B. D. and Hrishikesh D. Vinod (2003). "Verifying the Solution from a Nonlinear Solver: A Case Study". In: *American Economic Review* 93.3, pp. 873–892.

References III


Oreopoulos, Philip (2006). “Estimating Average and Local Average Treatment Effects of Education when Compulsory Schooling Laws Really Matter”. In: *American Economic Review* 96.1, pp. 152–175.


– (2008). “Estimating Average and Local Average Treatment Effects of Education when Compulsory Schooling Laws Really Matter: Corrigendum”. Available at http://dx.doi.org/10.1257/000282806776157641.


Rothstein, Jesse (2007a). “Does Competition among Public Schools Benefit Students and Taxpayers? Comment”. In: *American Economic Review* 97.5, pp. 2026–2037.


– (2007b). “Rejoinder to Hoxby”. Available at http://gsppi.berkeley.edu/faculty/jrothstein/hoxby/rejoinder.pdf.


Sweigart, Al (2015). *Automate the Boring Stuff with Python*. San Francisco, CA: No Starch Press.

Acknowledgements

- This course is designed after and borrows a lot from the `Software Carpentry` course designed by `Greg Wilson` for scientists and engineers.

- The Software Carpentry course material is made available under a `Creative Commons Attribution License`, as is this course's material.

License for the course material

[Links to the full `legal` text and the source text for `this` page.] You are free:

- **to Share** to copy, distribute and transmit the work
- **to Remix** to adapt the work

Under the following conditions:

- **Attribution** You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).

With the understanding that:

- **Waiver** Any of the above conditions can be waived if you get permission from the copyright holder.
- **Public Domain** Where the work or any of its elements is in the public domain under applicable law, that status is in no way affected by the license.
- **Other Rights** In no way are any of the following rights affected by the license:
  - Your fair dealing or fair use rights, or other applicable copyright exceptions and limitations;
  - The author's moral rights;
  - Rights other persons may have either in the work itself or in how the work is used, such as publicity or privacy rights.

Notice For any reuse or distribution, you must make clear to others the license terms of this work. The best way to do this is with a link to `this` web page.