An open-source model for training in reproducible data science

Background

We conceive data science very broadly, as a transformation in the role of code in data analysis.

Thus conceived, data science is a fundamental skill to analyze and understand data that applies across all quantitative fields.

It is a fundamental skill for research.

Many researchers use code, but few have training in what we call organized coding practice (OCP). OCP is standard in industry, and in open source projects. It has evolved in industry as programmers have learned how to better learn, collaborate, and avoid error. Without OCP, programmers all but invariably produce poorly organized code that is difficult to read, to debug and to share with others. It has a high density of errors, and inhibits further learning.

The elements of OCP are all but universally agreed; code review, continuous integration testing, version control, process automation with scripting.

Reproducible science is an essential reform in research practice, but as we have argued elsewhere [1], rigor and clarity are hard to retrofit. Among its many advantages for scientific code, OCP leads to reproducibility *by design*, reproducibility from the root.

Berkeley's huge and successful data science programme, and our own teaching, has shown that is it possible to teach data analysis through coding to all life science students, and students beyond the life sciences.

The principle of such courses has been given as "coding to learn" rather than "learning to code." Code is a way of expressing the logic of algorithms that might previously have been expressed in equations. Experience suggests that code makes it radically easier to see the ideas behind statistics and machine learning. This is an long argument made by, among others, by experienced teachers of statistics [3].

"Coding to learn" forms a basis for clearer exposition of many fundamental ideas in applied research, such as optimization, Fourier transforms, non-linear effects, multiple regression, to give select examples among many.

We have shown that is it possible to teach OCP to undergraduates and master's students with a limited previous knowledge of code, so that students are able to use all the standard elements of OCP to collaborate on a large open-ended data analysis project [1].

Our broad conception of data science holds that all researchers doing quantitative research should have:

- The ability to write code to load, explore, clean and analyze data.
- Training in OCP for research code, for reproducibility by design.

We will build on these foundations with “connector” courses that use “coding to learn” and OCP to improve exposition of more advanced ideas in specific research fields, and areas of study, while continuing training and reinforcement of reproducible research practice.

The problem

We believe the combination of “coding to learn” and training in OCP to be the correct basis for all training in quantitative research.

However, we have very few teachers who have experience of teaching coding-to-learn. As Berkeley have shown, and we have confirmed, coding-to-learn is a radically different approach to the more standard learning-to-code. It involves a much more focused approach on code for analyzing and visualizing data. There are very few teachers who have experience of this approach.

This applies even more to teaching in OCP. Although it is possible to teach OCP, even to undergraduates [1], this training continues to be tragically rare in research. As a result, reproducibility becomes an after-the-fact addition to otherwise disorganized coding practice, causing great waste of effort, and much higher chance of error.

We need some way to make this teaching scale.

Our approach to this problem is to use agile open-source process to build a community of contribution for such teaching materials. There are many obvious examples, but the fundamental libraries of scientific Python are particularly relevant. Recent descriptions othershow the astonishing productivity of small groups of highly committed contributors in building software that is displacing products supported over many years by large well-funded corporations.

We aim to replicate this approach in teaching. But there are barriers that we have to overcome:

- Common contribution needs material with an license that is open for editing and redistribution. Although the Berkeley materials are of outstanding quality, their license does not now editing and redistribution without explicit permission of the authors.
- The community for such contribution does not exist yet; we need to build it.
- Open source depends heavily on lowering the barrier to entry, and to contribution. To do this we need machinery that makes it simple to build lessons and exercises. It should be simple to edit these materials, and simple to give back the changes for others to use. There are tools to do this, but they are not yet widely used.
- A central tenet of open source practice is release-early, release-often. It is therefore absolutely essential that the process of releasing, or deploying teaching materials, should be as simple and painless as possible. This is not so, at the moment. Even deploying courses on Open edX, an open learning environment, is not straightforward, and it is particularly complex to integrate Open edX with the in-browser coding model that has been so important to code-to-learn courses such as those in Berkeley.

We will build:

- A library of teaching materials with a fully open license.
- A community of teachers that use and contribute to these materials.

- A system that supports contribution and redistribution.
- Mechanisms for simple default deployment of courses with integrated in-browser coding.

What we will teach

Will will first build and refine two **core** courses.

- **Foundations of data science** (FDS): code as foundation for data analysis, statistical inference and machine-learning.
- **Reproducible and collaborative data science** (RCSDS) : use of standard OCP for substantial, collaborative data analysis.

We will build on this foundation with connector courses that use the pedagogy and process from the core courses. The connector courses will cover:

- Subject-specific applications and extensions of core courses. For example, we will apply FDS and RCSDS methods for teaching computational social science, and biostatistics.
- Further depth in specific topics, such as probability, multiple regression, multiple hypothesis testing.

We will first assemble the materials from our current courses, and rewrite material that cannot reasonably be licensed for open contribution.

Following open-source method, we will refine our courses by rapid iteration. We will use the connector courses to refine our teaching in the core courses, and vice versa.

Plan

Our proposal has two key aims. The first is to build teaching materials and machinery round these materials to foster collaboration and rapid iteration.

The second is to build a community of teachers around these materials.

To further both these aims, we will iterate rapidly, while teaching across a wide range of research and teaching experience. We start by teaching our materials and pedagogy to established teachers and researchers, to build the community of contributors, introduce this wider conception of data science, show the pedagogy, and begin the process of training the trainers.

We will continue this training, while using the experience and contribution from these courses to refine our first courses to selected post-graduate researchers.

Next, we will recruit for full-semesters of open access to our core courses, while piloting our connector courses.

- Feb - May 2021 : collate, curate and rewrite (prepare) existing materials for core courses.
- June 2021 : two iterations of two-day “train the trainer” course, introducing topics from FDS and RCSDS.
- July 2021 : start collaboration-time grant process. Teachers apply for contributions from our team in the form of two or more weeks of focused team collaboration and contribution to connector courses or improvements to core courses.
- July 2021 : week-long summer camp form of pilot FDS course for junior researchers.
- August 2021 : week-long form of pilot RCSDS course for junior researchers, with prerequisites from prior coding experience or FDS course.

- September - December 2021: prepare materials and machinery for longer form blended FDS and RCSDS courses. Prepare materials for short and long form connector courses.
- January - May 2022 : deliver first iteration of blended long-form FDS course. Pilot blended components from connector courses.
- June - August 2022: multiple iterations of two-day “train the trainer” course, in the form of “roadshows” focused on particular areas or universities, sponsored by local host.
- September - December 2022: long-form blended versions of FDS; RCSDS, selected connector courses.
- January - February 2022: focus on collation and refinement of connector courses. Start repeat cycle of long-form FDS and RCSDS with wider range of connector courses.

References

1. Millman KJ, Brett M, Barnowski R, Poline J-B (2018) Teaching computational reproducibility for neuroimaging. Frontiers in Neuroscience
2. Cobb GW (2007) The introductory statistics course: A ptolemaic curriculum? Technology Innovations in Statistics Education 1:
3. Cobb G (2015) Mere renovation is too little too late: We need to rethink our uwillnder-graduate curriculum from the ground up. The American Statistician 69:266–282