How can we teaching coding to scientists

Matthew Brett

March 15, 2018

Plan of the talk

- why should we teach scientists to code?
- what should we teach them?
- how should we teach them?
- how can I help?

Why should we teach scientists to code

- Algorithms are at the heart of science. All scientists need to use and understand algorithms;
- Many branches of science use extensive computation.

Is it reasonable to teach scientists about algorithms?

- understand what algorithms are, how they are implemented as programs on digital devices, and that programs execute by following precise and unambiguous instructions;
- create and debug simple programs;
- use logical reasoning to predict the behaviour of simple programs
- use technology purposefully to create, organise, store, manipulate and retrieve digital content

Does anyone recognize this?

Understanding algorithms

- understand what algorithms are, how they are implemented as programs on digital devices, and that programs execute by following precise and unambiguous instructions;
- create and debug simple programs;
- use logical reasoning to predict the behaviour of simple programs
- use technology purposefully to create, organise, store, manipulate and retrieve digital content

National curriculum key stage 1: ages 5 to 7.

Algorithms for data analysis

It's very common to have the following situation:

- we have some somewhat messy data;
- we have some hypothesis about these data we would like to test;
- the hypothesis can often be formulated as a model.

The combination is - data science, and statistics.

Teaching data science, and all of statistics

Loading data, reviewing data, plotting data:

- Course page at https://matthew-brett.github.io/les-pilot;
- Our notebooks at https://notebooks.azure.com/matthewbrett/libraries/dev-ac-uk

With thanks and a recommendation for:

- Statistics without the agonizing pain:
- https://www.youtube.com/watch?v=5Dnw46eC-0o

Setup for the demo - running on Azure

- Go to https://notebooks.azure.com/matthewbrett/libraries/dev-ac-uk;
- Clone that site (you'll need a free Microsoft login), and click on the `getting_started` notebook.

Setup of the demo - running on your laptop

You can also run this on your own machine if you have Python and pip set up:

```
pip install pandas
pip install jupyter[notebook]
```

Download https://github.com/matthew-brett/dev.ac.uk (via ‘Clone or Download’, ‘Download zip’).

```
cd directory_containing_download
unzip dev.ac.uk-master.zip
cd dev.ac.uk-master
jupyter notebook
```

Code for everyone

We have seen code as a way of:

- expressing operations on data;
- doing (and explaining) statistical inference.

Superficially, this is a course for scientists. But all academic fields are now analyzing data, as are the media. So, how should we teach?

Berkeley Fundamentals of Data Science course

- All subjects (arts, sciences . . .).
- Written as interactive Jupyter Notebooks.
- Textbook at https://www.inferentialthinking.com

Back to the demo

- https://notebooks.azure.com/matthewbrett/libraries/dev-ac-uk
- `Plotting the classics.ipynb`.
- `Literary characters.ipynb`.

See the README for the site for the license of these files.

The Berkeley course is substantial

See :
https://www.inferentialthinking.com/chapters/16/classification.html

Beyond the basics

- Many academic fields need a lot of computation.
- Computation is often done in a very sloppy and disorganized way.
- It's easy to make errors.
- People usually don't share their analyses.

Easy to make errors in computation

> *In my own experience, error is ubiquitous in scientific computing, and one needs to work very diligently and energetically to eliminate it. One needs a very clear idea of what has been done in order to know where to look for likely sources of error. I often cannot really be sure what a student or colleague has done from his/her own presentation, and in fact often his/her description does not agree with my own understanding of what has been done, once I look carefully at the scripts.*

David L. Donoho (2010) "An invitation to reproducible computational research" *Biostatistics* 11(3) 385

Sharing analyses

> *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.*

Jonathan B. Buckheit and David L. Donoho (1995) "Wavelab and reproducible research"
http://statweb.stanford.edu/~wavelab/Wavelab_850/wavelab.pdf

What tools do we need to teach for this endeavor?

What tools do we need to teach for this endeavor?

- Unix command line;
- Text editor for everything (including presentations);
- Git for version control;
- A scientific programming language such as Python or R;
- Unit testing, code coverage, continuous integration;
- make / Makefiles;

Can it be taught?

- http://www.jarrodmillman.com/rcsds
- https://github.com/berkeley-stat159

How can I help?

- It's a culture; spread the culture;
- Organize meetups, interest groups, talks;
- Find like-minded lecturers to support you, and for you to support.
- Consider forming or joining a Hacker Within group: http://www.thehackerwithin.org.

Is this the end?

Yes, it's the end of the talk.