Where will data science go?

The term *data science* has come to stand for many different *movements* in science, all loosely related to computing. I believe data science is itself a movement, or a culture, rather than a branch of learning. If it is a movement, where are we in the trajectory of that movement, and where will we end up?

Movements within "data science"

Changes in the way we think about software

The way that we think about software has changed radically in response to the success of free / open source software (F/OSS).

The rise of free / open source software

Twenty years ago, the term "software" meant something that came in a shrink-wrapped box, written by a large and expensive software development team, and whose inner workings were opaque. Both the software, and the ideas behind the software, belonged to the company that produced it. Since then, the F/OSS movement has produced a large range of high quality software, of which Linux, Python and the R statistics language are three prominent examples. F/OSS software can be produced by relatively small teams of distributed programmers, working without pay; their work is transparent rather than opaque, public rather than private.

Open source and the scientific method

As it became clear that software *could* be written effectively using F/OSS, it also became clear that the principles of F/OSS matched those of the scientific method. F/OSS, like science, embraces transparency as a means to sharing information and correcting error.

Agile development and code quality

Over the same period, there has been a large change in standard software development process, from the standard corporate "waterfall" method, where the development proceeds in specialized stages, such as "requirements analysis" and "architecture", to the new standard "agile" or "extreme" programming methods, where small multi-skilled teams work in close collaboration throughout the project. Agile methods have clear parallels to Toyota's "lean" production methods, with close attention to improving process to optimize for productivity and quality.

The agile movement led to many developments in working practice, such as code review, pair-coding, advances in automated code testing, and version control to record code history. All these techniques have diffused rapidly into open-source, and thence into scientific software development. Their particular emphasis is on quality and productive collaboration, and they have made it much easier to develop and maintain high-quality scientific code.

Code is method / take back the code

Over the last twenty years, most scientists have learned to write code. As they have done so, it has become clear that *code is method*. That is, that the algorithms instantiated in the code are the fundamental stuff of data analysis. Many algorithms rely on careful handling of corner cases and quirks, which are difficult to specify outside the code. The code is the definitive and often the clearest guide to the algorithm.

The combination of improved development practice and models of F/OSS success showed scientists that they must recover control and understanding of their analysis by writing their own code, and contributing to code they rely on.

Experience gives greater awareness of error

It is characteristic of inexperienced developers that they make more errors, are less likely to detect errors, and underestimate the number of errors they are making. To quote Charles Darwin: "Ignorance more frequently begets confidence than does knowledge". This is a dangerous situation if the code is critical to the outcome, as is often the case in science. When few scientists were experienced developers, it seemed reasonable to assume that scientists, on the whole, were producing code that did a correct analysis on the data it would be applied to. Now we have more experience, we know that this assumption is very unlikely to be correct, and that we have to train our students to stop them falling into the same complacency that we suffered ourselves, early in our careers.

Example: the rise of open source statistics

We can see the sum of the factors above in the history of statistical software. Twenty years ago, a researcher using statistics would likely choose from a number of large expensive software packages such as BMDP, SAS, SPSS, and S plus. Statisticians developing new methods had to chose which of these packages they would develop for. Users could only practically use methods developed for the package they had access to.

In the early 90s, two statisticians from New Zealand wrote a F/OSS version of the S plus language, called R. It has rapidly become the *de facto* standard statistical package used and taught by statisticians. It is common for new methods to be released with R code to implement them.

Changes in the way we teach programming

There appears to have been a significant shift in teaching of programming, both at school and university. Schools now do earlier and more substantial teaching on hardware, code and algorithms. In the new UK computer science curriculum 'Pupils aged five to seven will be expected to "understand what algorithms are" and to "create and debug simple programs."'[1] The $35 Raspberry Pi computer was designed to teach children in secondary school how to understand computers and program them, where the "Pi" is an explicit reference to Python, the main language used for coding on these machines. The current edition of the official "Raspberry Pi User Guide" has an introduction starting with a section "Programming is Fun!", but chapter 3 (of 16) is "Linux System Administration" and has sections on "File System Layout" and "Managing Software at the Command Line."[2]

In universities, introductory courses in programming are now more likely to be taught in Python than other languages such as Java (perceived as attractive for industry) or Lisp / Scheme (admired for their purity of syntax and design). At the same time, Python has advanced as a language for scientific computing, with great expansion in the range and quality of general scientific and domain specific libraries. The spread of Python into introductory and research computing means that students start practical data analysis with a programming language that they can use to analyze their data. They have the tools they need to implement any algorithm they can think of. They may already have specific Python libraries for their own domain. This changes what the student will do and how they think about data analysis.

As Python has become standard for scientific computing, it has started to change the way that we teach. When the student understands simple code in a language like Python, it is often revealing to link the underlying mathematics to

[1] http://www.bbc.co.uk/news/education-23222068

[2] Raspberry Pi User Guide 4th edition (2016) Eben Upton and Gareth Halfacree. Wiley

its implementation. Students can understand mathematics through the code, and explore the algorithms used for the analysis by reading and modifying the analysis code.

A wider range of algorithms for data analysis

Developments in computer science and statistics have caused considerable shifts in the types of analysis that researchers are using. In particular, the success of various forms of prediction and classification algorithms often categorized as "machine learning" have allowed researchers to ask a broader range of questions of their data.

An increase in data volume

There has been a vast increase in our ability to collect, store and analyze data, driven by a combination of improved hardware for data collection and storage, and improved software for parallel computation and distributed storage. Using and writing such software is a major challenge for scientists without training in unfamiliar subjects such as parallel programming and system administration.

Symptoms of these movements

Summarizing:

- rising importance of researchers competent in programming, and the lack of career support for such researchers in a world increasingly driven by publication metrics;
- increasing appreciation of the importance of scientific software to science in general, and to individual labs, and the lack of support for such software in current funding models;
- increasing numbers of a new type of researcher, competent in programming, that uses programming for data analysis, using a combination of new code and existing code libraries, typically in Python or R - the "data scientist";
- the rise of the reproducible software movement, as open source code becomes the norm, and working practice improves to make this practical.

Responses

Summarizing:

- "data science" institutes (e.g, Berkeley, Harvard). The argument here is that there is benefit to bringing together scientists that are using similar

methods and / or tools, in order to collaborate across domains. A centre may make it easier for domain experts to find computational experts willing to advise or collaborate. The university will benefit from supporting researchers who implement and teach software for data analysis;

- computational teaching - often called "data science". Example, UC Berkeley data science main courses and connector courses. The argument is that students should learn to explore data using code, to give them deeper understanding of the limits of analysis, expand the range of analyses they can choose from, and improve their understanding of standard analysis methods.
- research software engineers - see[3] - a UK-led initiative to support the subset of scientists who want to work on software full- or near full-time.

Where is data science?

I suggest that we are now at a stage where the relationship of the various movements above is starting to become clear. I believe that they can be summarized into three primary themes:

- an overwhelming pressure, from research, industry, and our students, to teach data analysis and inference using code;
- a recognition of the central place of code in scientific method - *code is method* - and that scientists must understand, contribute to and write analysis code - *take back the code*;
- a nascent revolution in the process of scientific inquiry, driven by the influx of agile / lean methodology from software development via open source.

Where will we go?

We can answer the question of where we are with reference to the end point of these movements above. I think we are going to see the following:

- it will be standard to teach analysis methods using code in a high-level language such as Python;
- increased familiarity with code and data analysis will lead to a rapid diffusion of new data analysis methods across disciplines;
- widespread recognition of the central role of software in data analysis. Researchers and students will engage directly with software - they will be strict in choosing F/OSS over alternatives, they will write their own code, contribute code to the libraries they use for their analysis, and spend some proportion of their time helping with maintenance of these libraries;

[3] State of the Nation report on Research Software Engineers

- increasing experience with software will lead to universal adoption of "best practice" in scientific development (Wilson et al 2014), leading to increases in scientific productivity and quality;
- greater contact with agile / mean process in software development, and an increasing emphasis on software development in scientific training, will lead to greater adoption of adapted agile / lean methods for scientific projects, such as small multi-skilled teams, short plan-implement-review feedback cycles, and close collaboration between team members with different expertise. There will be an increasing emphasis on improving scientifi working process, to increase the efficiency and quality of research.

comment: () (S: Industry has more or less figured out lean/agile as applied to their interests. Significant modification may be needed for the adoption into science; we don't have many experiments to inform that redesign.)

Bibliography

Donoho, David. 2015. "50 Years of Data Science." In *Princeton NJ, Tukey Centennial Workshop*. http://courses.csail.mit.edu/18.337/2015/docs/50YearsDataScience.pdf.

Tukey, John W. 1962. "The Future of Data Analysis." *The Annals of Mathematical Statistics* 33 (1). JSTOR:1–67. http://projecteuclid.org/euclid.aoms/1177704711.

Wilson, Greg, DA Aruliah, C Titus Brown, Neil P Chue Hong, Matt Davis, Richard T Guy, Steven HD Haddock, et al. 2014. "Best Practices for Scientific Computing." *PLoS Biol* 12 (1). Public Library of Science:e1001745. https://doi.org/10.1371/journal.pbio.1001745.