In a recent blog post, Titus Brown asks if software is a primary product of science, and basically says "no" (but do read the post for the details). A blog-post length reply by Daniel Katz comes to the opposite conclusion (again, please read the post before continuing here). I left a short comment on Titus' blog but also felt compelled to expand this into a blog post of its own - so here it is.

Titus introduces a useful criterion for what "primary product of science" is: could you get a Nobel prize for it? As Dan comments, Nobel prizes in science are awarded for discoveries and inventions. There we no computers when Alfred Nobel set up his foundation, so we have to extrapolate this definition a bit to today's situation. Is software like a discovery? Clearly not. Like an invention? Perhaps, but it doesn't fit very well. Dan makes a comparison with scientific writing, i.e. papers, textbooks, etc. Scientific writing is the traditional way to communicate discoveries and inventions. But what scientists get Nobel prizes for is not the papers, but the work described therein. Papers are not primary products of science either, they are just a means of communication. There is a fairly good analogy between papers and their contents on one hand, and software and algorithms on the other hand. And algorithms are very well comparable to discoveries and inventions. Moreover, many of today's scientific models are in fact expressed as algorithms. My conclusion is that algorithms clearly count as a primary product of science, but software doesn't. Software is a means of communication, just like papers or textbooks.

The analogy isn't perfect, however. The big difference between a paper and a piece of software is that you can feed the latter into a computer to make it do something. Software is thus a scientific tool as well as a means of communication. In fact, today's computational science gives more importance to the tool aspect than to the communication aspect. The main questions asked about scientific software are "What does it do?" and "How efficient is it?" When considering software as a means of communication, we would ask questions such as "Is it well-written, clear, elegant?", "How general is the formulation?", or "Can I use it as the basis for developing new science?". These questions are beginning to be heard, in the context of the scientific software crisis and the need for reproducible research. But they are still second thoughts. We actually accept as normal that the scientific contents of software, i.e. the models implemented by it, are understandable only to software specialists, meaning that for the majority of users, the software is just a black box. Could you imagine this for a paper? "This paper is very obscure, but the people who wrote it are very smart, so let's trust them and base our research on their conclusions." Did you ever hear such a claim? Not me.

Scientists haven't yet fully grasped the particular status of software as both an information carrier and a tool. That may be one of the few characteristics they share with lawyers. The latter make a difference between "data" (including written text), which is covered by copyright, and "software", which is covered by both copyright and licenses, and in some countries also by patents. Superficially, this makes sense, as it reflects the dual nature of software. It suffers, however, from two problems. First of all, the
distinction exists only in the intention of the author, which is hard to pin down. Software is just data that

can be interpreted as instructions for a computer. One could conceivably write some interpreter that
turns previously generated data into software by executing it. Second, and that’s a problem for science,
the licensing aspect of software is much more restrictive than the copyright aspect. If you describe an
algorithm informally in a paper, you have to deal only with copyright. If you communicate it in
executable form, you have to worry about licensing and patents as well, even if your main intention is
more precise communication.

I have written a detailed article about the problems resulting from the badly understood dual nature of
scientific software, which I won’t repeat here. I have also proposed a solution, the development of
formal languages for expressing complex scientific models, and I am experimenting with a concrete
approach to get there. I mention this here mainly to motivate my conclusion:

- Q: Is software a primary product of science?
- A: No. But neither is a paper or a textbook.
- Q: Is software a means of communication for primary products of science?
- A: Yes, but it’s a bad one. We need something better.

Explore posts in the same categories: Computational science