How to become a programming tadpole

James G.R. Gilbert

Beginning Perl for Bioinformatics
By James Tisdall
O'Reilly & Associates; $39.95, 368 pp, paperback, ISBN 0-596-00080-4, 2001

Biologists these days can generate vast quantities of data with relative ease. Bioinformaticians write the programs that facilitate the production and use of this data. It is possible get by using the many excellent bioinformatics tools, such as Blast, without having to learn to program. But researchers may find themselves performing a repetitive task, or scanning through large amounts of output by eye for a particular pattern. A little programming knowledge can be very useful in automating these tasks.

Programming can seem an arcane activity to biologists. The field abounds with jargon; the computing shelves of book shops display a bewildering array of choices; and computer professionals have very strongly held but contradictory opinions about which language would be the best to learn. So where to start? I think (speaking as a computing professional with strongly held opinions) *Beginning Perl for Bioinformatics* by James Tisdall would be a great place. Aimed at biologists who are complete novice programmers, he covers the features of the Perl programming language that will be of most use in helping solve biological problems. The examples used contain code that is actually useful. He shows how to reverse-complement DNA, simulate random mutations, find restriction enzyme–binding sites, translate RNA to protein, and parse data out of GenBank, PDB, and Blast reports. This is a refreshing change from computer language references where the practical examples are invariably personnel records!

Perl is a language that is particularly suited to bioinformatics. It began as a tool for text manipulation and report generation. It grew to include facilities for dealing with data stored in databases, vital for the vast data sets we have today. It has some of the best tools for communicating over the Internet, presenting graphical views of data, and is great for gluing lots of uncooperative programs and databases together to produce automated systems. In addition, Perl is free, open-source software. You can write code and distribute it without fear that you will run into problems caused by a restrictive software license.

As the author says, "Perl has a long and low learning curve". It is low because using Perl it is easy to produce little programs that do useful work early on in the learning process. One of the first things I ever needed to program, back in the days when I still occasionally wielded a pipette, was to reverse-complement a large (more than 80 kbp) BAC of mouse genomic sequence that The Sanger Centre had sequenced for us. The DNA analysis program on our Macintosh was unable to cope with anything longer than 64 kbp, so I wrote a Perl script on the University UNIX machine we used for our email. An example of how to do this is one of the early ones in the book. The learning curve is long, too. Despite my five years experience with the language, I still picked up a couple of tricks from this book.

Tisdall takes time to explain how to go about writing software, how to think about the design of your program, and then implementing it. This is a nice ingredient that is missing from many instruction books. His teaching experience is evident from the traps for the novice programmer that he highlights. Perl syntax is particularly flexible—a Perl motto is "There's more than one way to do it". This can make Perl code confusing for the novice, but he has been careful to use consistent idioms throughout the book. Optional features of the language (such as warnings and the "strict" pragma) that trap a lot of common Perl programming errors are emphasized. Such advice would certainly have saved me a lot of wasted time when I was learning the language.

O'Reilly continues to publish the definitive guides to this programming language. And they produce some of the most readable and witty computing texts. The example code they contain is carefully checked for errors, which are particularly frustrating in computing books. The covers of most of their books feature a distinctive woodcut of an animal; in this case three frog tadpoles. If you begin reading as a programming "egg", and work your way carefully through the exercises, then the tadpole will be a good representation of your stage of development by the end. There is plenty more to learn about Perl, some of which is touched on in the last chapter. Object-oriented programming techniques are not taught, which are needed for using most of the publicly available Perl libraries. I imagine that part of the problem of writing an introductory book like this was deciding what to leave out.

James G.R. Gilbert is at the Wellcome Trust Sanger Institute, Hinxton, Cambridge CB10 1SA, UK (jgrg@sanger.ac.uk).