TEACHING STATEMENT

SCOTT SHERRILL-MIX

The expansion of high throughput methods have revolutionized biology. However, with this big data comes computational problems. Upcoming biologists are often hindered by a lack of computational understanding and abilities. An understanding of statistics, ata analysis and bioifnormatics is an essential tool for modern biologists and students need to achieve competency in these areas.

I believe that one of the key problems in developing bioinformatic skills is the development of a mental model of computatino. Some take to programming and enoy. Some find arduous arcane task. Some enjoy but difficult progressing.

During my graduate training, I served as a teaching assistant for several biology and programming classes. I found the most enjoyable aspects of teaching were guiding students through interactive labs, both computational and biological. Recently, I have continued this hands-on approach by mentoring several colleagues from never having programmed into competent coders.

Across these experiences, I have seen that lessons on programming and bioinformatics are not well retained if the student does not have both an applied problem to solve and a mental model of the computation. For researchers interested in developing these skill, one course I would like to develop would focus on self implementation of common bioinformatic algorithms. Such a foundational course would demystify bioinformatics while offering the opportunity for teaching best practices such as documentation, software testing, source control and reproducible research. Adding small competitive programming challenges to the class might help to make learning to program fun and increase retention.

It seems many biologists view programming and statistics classes as uninteresting prerequisites to be rote memorized. My most memorable classes used real world examples and applied problems to stimulate student interest. For example, separating out "elite" baseball players or classifying the gender of portraits makes a much more interesting introduction to Bayesian statistics and machine learning. Another way to introduce interest in students is to add a small amount of competition. For example, running a "hackathon" where groups of students seek to build the best performing model in a machine learning task. This gamification of an applied example allows students to practice theory and allows direct enforcement of best practices by requiring students to submit through version control to automated testing and evaluation. In this vein, I have organized several student "hackathons" which combined a fun coding competition with experience in collaboration and programming.