The End Parenthesis: How a Fisheries Student Tackled Programming

Article *in* Fisheries · December 2016

DOI: 10.1080/03632415.2016.1252624

1 author:

Andrew Shamaskin
Mississippi State University
2 PUBLICATIONS **0** CITATIONS

SEE PROFILE

The End Parenthesis: How a Fisheries Student Tackled Programming

Andrew Shamaskin

It was 7 p.m. on a Wednesday, and I found myself planted by my desktop, finishing a computer program designed to generate sampling schedules for creel surveys. After making a few more edits, the code ran perfectly. At that point, I decided my eyes have had enough screen time for the day, so I saved my work and went home to rest up. The next morning, I opened the program and ran it a second time before sending it to my advisor. Although computers are designed to conduct processes the same as before, checking my code only twice still puts me one short of my benchmark for insanity. The purpose of rerunning my already-verified code was nothing more than a just-in-case tactic. I like to call this strategy a "courtesy flush." But even with my absolute certainty that the program will still work, the "courtesy flush" returned two errors. What began as a mundane Thursday morning, quickly transitioned back to a painful repeat of yesterday.

I'll return to my story, but I should first explain what I do, and how I found myself programming computers for my master's instead of getting to squeeze fish like the rest of my lab mates. My research assistantship at Mississippi State University focuses on length-limit modeling, and building software for the Fisheries Bureau of the Mississippi Department of Wildlife, Fisheries, and Parks to handle their data-entry-and–analysis needs. While I appreciate the depth of my studies, I always enjoy the exposure to broader knowledge from other students. One of my favorite aspects of working in my department is the diversity of projects in which my peers are involved. I love asking the other graduate students about their research, and likewise sharing my work with them. While our job distinctions can create separation, we find unity in the methods and tools used to conduct our research. Computers are one such tool fundamental to graduate assistants, as is the case in many occupations today. However, many of my cohorts would agree that it was our proclivity for the outdoors, and not a desire to learn R or PYTHON, which led us to where we are today.

The irony of the previous sentence is that I knew my assistantship involved no field work when I accepted it. I saw the tradeoff being an enhanced opportunity to build my chops in data analysis and mathematical modeling. Having challenged myself with a few extra quantitative courses as an undergraduate, I figured this project would refine my use of math within the context of fisheries. The surprise came on my first day when I realized the amount of software development my position would involve. Before my time at Mississippi State, the only encounter I had with programming was through an undergraduate roommate, who would sometimes make up words to express frustration towards his computer. Even though I had previously used SAS and R in my statistics classes, building programs was still an enigma, as my professors would furnish prewritten code for us to modify. My foundational knowledge coming into graduate school was firmly rooted in the biological, chemical, and ecological realms. The essential concepts made sense to me, and all my time spent outside further ensured retention of that knowledge. Back in high school, I recognized that I had a visually-dominant learning

strategy, and although I never struggled with mathematics or other logical-thinking coursework, I usually felt uncomfortable with the material unless I could easily frame it in a graphic. I persevered through the abstract concepts almost solely on rote memorization. It remained, however, that if I couldn't picture what was happening, then it was next to impossible to comprehend. I resolved to tackle this weakness. After I declared my B.S. in fisheries science at Virginia Tech, I used the remainder of my elective credits in the math and statistics departments. But I also knew my previous strategy toward quantitative classes wouldn't suffice; I needed to think uninhibitedly.

Along with a little perseverance, that mental restructuring gave me the tools to think mathematically. It began to click when I reviewed problems holistically at first and then broke them down into segments. This approach highlighted the contributions of each component within an equation, making problems much more digestible. When I could identify and understand the overall objective, I could then rationalize what steps I needed to take to reach it. That same principle applied to my perception of computer programming. A single line of code made no sense unless I could see it as a step towards a bigger goal. My journey through programming thus far has taught me proficiency in multiple computer languages, and every day I recognize the new potential in what my code can accomplish. It's no secret contemporary fisheries research employs significant computational power, which is often necessary to conduct robust analyses promptly. But I've also developed a far more subtle understanding of this technology as a utility in fisheries, and other fields within natural resources.

The direct purpose of programming is to tell a computer to carry out some task. Telling the computer what to do to accomplish the task starts with making a list of steps, then translating that list into a programming language that the computer can understand. Computer programming can allow the user to simplify complex or tedious processes, which sometimes may not even be feasible to carry out manually. In the case of fisheries science, automating processes can allow us to ask a computer questions about our data, and enhance our understanding of what our data can show us. But programs can go beyond human-to-computer communication. Programming is also a means with which people can talk to other people, and explain ideas and information in more effective manners. For my thesis, I'm exploring new ways to evaluate length limits and am encapsulating the process in the form of computer applications that are simple to use. Through these applications, I can communicate how I look at length-limit regulations to managers, and to stakeholders. These programs allow the user to interact with the length-limit model, input any scenario, and gain a better understanding of how length limits can affect fisheries. In every fisheries meeting I've attended, speakers echo a perpetual need to improve relations with our stakeholders, and our ability to communicate with them is at the core of that relationship. With programming, we can create a unique interface between fisheries scientists, fisheries managers, and the public, which can aid in building those valuable relationships. Creating tools to communicate our science and allow managers to demonstrate to the public how we determine our actions can not only improve their understanding and trust of what we do, but can also give us a much stronger case for our relevance.

As for my Thursday morning, the program finally worked after I recognized a missing parenthesis within the depths of my code, causing every other step to misfire. I must have accidentally deleted it the night before, while I was saving my files. This lesson showed me the peculiarities of computer technologies, and despite how intricate and intimidating they may appear, learning to utilize these tools is rarely as hard as it seems. On occasion, errors puzzle even the most seasoned programmers, but sometimes the solution is as simple as taking the lens cap off.