The Soft Approach to Software

By Simon Boxall

As COVID restrictions fade, though not the pathogen itself, universities are returning to a more normal way of life: live lectures, laboratory exercises, and face-to-face meetings with colleagues. Shortly before Christmas, we had our first post-COVID ocean physics teaching group meeting, a work/social event, in a local pub. Such gatherings in the past would have started late afternoon and lasted well into the early hours. Less so now, as members of the group are either getting too old (like myself) to keep up with the pace or now have young children, so the option of staying out late would, at best, be unpopular on the home front.

The work discussion drifted into computing and software, and how we teach the use of software for data processing and modeling to our undergraduates. This is a perennial issue and one that divides us very much along the discipline lines of marine science. In physical oceanography there is a strong push for MATLAB, the marine biologists support R, and the marine geologists go more for GIS packages such as ArcInfo. Each software tool has its own strengths and weaknesses, and each is suited to specific disciplines. The statistical packages for marine biology are well served in R, MATLAB has great large database and time series crunching ability, and ArcInfo is good for merging multivariate data sets. There are then questions about more fundamental programming, higher level languages, such as Basic, Fortran, and Python (and to an extent MATLAB), which allow interfacing with equipment and bespoke applications. There are simple graphics and data processing programs such as Excel, Sigmaplot, and Numbers, to name but a few. Finally, there is imaging processing software such as ENVI and PCI, and there are modeling programs like MIKE-21. This list, of course, is far from exhaustive.

None of these software packages offers a clear single front runner that will ready our students for the world of work. One key issue that came out of our very sober meeting was that, individually, we all use different packages and could not ourselves hope to teach more than one or two of them. It also seems the more senior you are, the less you use these packages on a regular basis—we all tend to send students with queries on software programming to our respective postgraduate students who use them on a daily basis.

As a postgrad, I learned Fortran on a mainframe computer, and I have to say I was pretty good (and so modest). I worked with it on a regular basis to process CTD data; there was nothing else available. As I progressed onto my first postdoc position, the use of personal computers came in and the languages grew to include Basic and COBOL. To say there were no off-the-shelf programs would be an understatement. When we wanted to interface a CTD with a computer, we had to write a program that would speak through the computer's comms port to the CTD and control data handling, storage, and display. Once data had been secured on the computer, producing graphs was yet another issue. Marine instrumentation rarely came with software back in the 1980s. The key elements and processes were similar whatever the programming language, and it never took long to switch between languages. As time progressed, Microsoft Windows came along, as did scientific equipment complete with interface and display software. The need to program diminished for everyday work, and while modelers were at one with their software, we observational oceanographers were happy with the packages provided. We moved onto higher level programs—in my case, image processing software and coastal modeling packages.

Early days of satellite and airborne data meant dealing with a very different data set—raster data (we just call them images or photos now). These once needed powerful computers with specialist and expensive software—the sort that was copyright protected by computer dongles so you could only run it on one machine for each license. I became competent (modesty kicks in here) at juggling image data, geo-correcting raw data, calibrating, overlaying multichannel data, and even exporting to GIS packages to merge with none-raster data sets. Today, dealing with image data is easy, software is available on most laptops, and the primary databases such as NASA and ESA provide all of the data in real time pre-processed to a high level—just look at the SOTO pages on the PODAAC site https://podaac.jpl.nasa.gov. The level and sophistication of data access available in minutes now would have required months in the past, to say nothing of the processing expertise once needed.

So, do we lose these skills with time? I don't think we do, but it does take a bit of a refresher to get back to where we were, and unfortunately it is not like the proverbial "riding a bike." An example I encountered recently was when a student

needed to use the coastal modeling program MIKE-21. Some years ago, I used this program extensively for a particular project, and the department did run a teaching license. We duly purchased a new teaching license for the student's dissertation and set it up on their computer. Some days later they came to see me as they couldn't work out where to start. In a rush of bravado and self-confidence, I rolled up my sleeves, smiled, and started to type—nothing. After about an hour of getting nowhere, I read the manual (had I been of a different gender, that would have happened 50 minutes earlier…). Even less the wiser having read it, I called up the support of my colleague who had taught the course dealing with it a couple of years earlier. She did resort to the manual after five minutes (it is a gender thing), but an hour later we still hadn't got off the launchpad. Hanging our heads, we went to the software provider, who quickly got us going. The difficulty was the result of a simple, but well-hidden, change in the most recent version that neither of us knew about. (Well, *we* thought it was well hidden.)

I liken it to our new departmental car that I drove for the first time a few months ago. I know how to drive (despite what my wife thinks). I know how to enter and start cars—or thought I did. But when you jump in a hybrid-keyless-electronic-hand-brake car, you find that starting is different—there is no "key" as such. I sat there having entered the car, looking for the place for the ignition key before realizing there was no key as such. Having found a start button by the cup holders I pushed it—waited while the car spent some time checking itself over and then waited again while the car waited for me to put my foot on the brake. It was a battle of wits, but safe to say the car won and I resorted to the manual. With no engine noises, I surmised the car was in electric mode and engaged reverse. I searched high and low for a handbrake and eventually realized that the car had decided it was time to go, and it didn't need me or a handbrake that I controlled. Once we were off, the car drove like a normal car, and it did need me. Software is a bit like that.

So, what software packages should we teach our students? If I speak to first-year students, I am pleasantly surprised when they can cope with Excel—not all can at first. As they progress to year two, by which time they will have sat introductory courses in R and MATLAB, they are confident in Excel as it seen as the easier option compared to its more complex counterparts. When asked if they use the more advanced packages, a look of horror and fear comes across their faces. Year three brings a new level with need to process more complex data sets. Some will now use MATLAB or R (never both) very successfully in their work. Some will thrive to take on Python without our help and are quick to convert their contemporaries. A few will discover the delights of Sigmaplot for plotting, as long as they have a Windows-based environment. The majority? Excel.

There was one student this year who didn't even get to these heady levels of technical achievement. They inquired whether they could submit hand-written reports—it appeared that they didn't like computers and preferred pen and paper. I asked whether they had a computer and if it was a Mac or PC. The reply was yes, they weren't sure, but it was silver. I then asked how they could manage calculating even the simplest of tasks like averages and standard deviations—the answer was on paper, even with many hundreds of data points, and that is why work was always submitted late. The other end of the spectrum are students who use software such as R to produce nonsense statistics without thinking through cause and effect. In work I marked last month, this included someone getting excited by the statistical relationship between pressure and depth from CTD data, and someone else who proved there was no correlation between the day of the week and plankton populations. Just because a button in the software lets you do it, you don't have to push it!

So, we concluded after our pre-Christmas get-together that academic staff themselves could do with software refresher courses before doing anything else. There is a need to encourage students to engage with programming in order to do more with the data they need to process, but many of the packages we use are very specific to research or certain applications. When they graduate and enter industry or academia, the programs used will be different, and they will receive training in those packages. What we need is the ability for a student to think a bit like a computer (in a positive way), but also to keep thinking as a scientist. Complex and expensive packages are not much use when a student graduates and has access to neither the support from an able postgraduate or access to the educational license. Software packages often fall into redundancy or are modified beyond recognition within a few years, so it is the basics of programming they need to grasp. Our conclusion was that we need to teach them Python, something we haven't done to date, and to learn it ourselves. We will have difficulty persuading our marine biology colleagues to move away from R, and we will all still use our various packages in our research, but if you ask the majority of undergraduates what they use today—number one is Excel, number two is Python. We now need to get them to teach us all Python. They do say the best way to learn is to teach.

AUTHOR

Simon Boxall (srb2@noc.soton.ac.uk) is Associate Professor, Ocean and Earth Science, University of Southampton, National Oceanography Centre, Southampton, UK.

ARTICLE CITATION

Boxall, S. 2023. The soft approach to software. *Oceanography* 36(1):76–77, https://doi.org/10.5670/oceanog.2023.114.