WRITE WAY Why a researcher left the lab to become a science reporter **p.565**

BLOG Personal stories and careers counsel **blogs.nature.com/naturejobs**

NATUREJOBS For the latest career listings and advice **www.naturejobs.com**

GETTY IMAGES

Programming languages give you much more freedom than you have with commercial applications.

SCIENTIFIC COMPUTING

Code alert

Programming tools can speed up and strengthen analyses, but mastering the skills takes time and can be daunting.

BY MONYA BAKER

Auriel Fournier had no choice but to learn programming. The ecology PhD student wanted to use a complex set of calculations to estimate migratory populations from field observations, and doing so efficiently required a software package that ran in the programming language R. Her principal investigator (PI) did not know the language. Neither did anyone else in her lab at the University of Arkansas in Fayetteville. "My PI said, 'Figure it out'," says Fournier. She began googling online tutorials, mastered the package and now helps other researchers to make sense of R and similar tools. It can be "really intimidating" to learn a programming language, but the long-term benefits are well worth the effort, she says.

Many commercial computer applications such as Word, Excel and Firefox are programmed for ease of use, at least for routine tasks, but those who can instruct computers directly can customize them to perform tasks that are more powerful and specific to their research. Working in commercial applications is limiting, says Sacha Epskamp, a methodologist and psychometrician at the University of Amsterdam. "You can only do what the buttons say you can do." Programming languages are much broader in scope, he says. "In R or other programming languages you can do anything you want." But to achieve this broad latitude, researchers need to learn how to code — to formulate commands in a programming language that tell a computer what to do. Researchers who know just a bit of code gain access to a wealth of software packages that automate repetitive tasks and increase options for compiling, analysing, sharing and presenting data.

Coders often use R for doing statistics, whereas Python, which is often considered more intuitive, is good for repetitive tasks and for extracting data from web-based applications. Ruby provides more ways of doing the same tasks and is popular with web developers.

Learning to code is empowering and can hugely improve a researcher's career prospects. "But it does require an investment," says Fournier. And that includes not just time, but also an ongoing effort to assemble a community of helpful experts and to be willing to make, find and fix mistakes.

SMOOTH OPERATORS

Coding is becoming a crucial part of research, says Ethan White, an ecologist at the University of Florida, Gainesville, who has designed courses in quantitative methods. "When you do this well, your life is easier and your results can be regenerated," he points out. Scientists commonly use languages such as Python and R to conduct and automate analyses, because in this way they can speed data crunching, increase reproducibility, protect data from accidental deletion or alteration and handle data sets that would overwhelm commercial applications. Researchers who use these languages can tackle questions that would be impractical to address if data were manipulated manually, says White. Reconfiguring an analysis or revising graphs becomes quick and straightforward, and researchers can more easily build on their own or others' work.

Andrew Durso can vouch for those upsides. The ecology graduate student at Utah State University in Logan started his research career using programs with graphical interfaces. Whenever he clicked buttons or checked boxes on a computer screen, he would try to write those steps down on paper in case he wanted to redo an analysis — a strategy that was both time-consuming and unreliable. Even though he is ▶

still new to coding, he says, a re-analysis of one of his earlier projects took less than 5% of the time previously required, and every step was recorded automatically.

GETTING STARTED

Some of the biggest barriers come at the start of the learning curve, says Karl Broman, a biostatistician at the University of Wisconsin–Madison. First, users have to figure out how to install or access a programming language. That requires getting to an interface known as the command line, where point-and-click no longer applies. (Imagine those black screens with scrolling green type in movies such as *The Matrix*.) Getting the hang of typing instructions is another hurdle. "People don't realize that coding is different from what you generally experience on your computer," says Marquitta White, a postdoc studying asthma at the University of California, San Francisco, who trains students to perform genetic analyses. A machine requires precise instructions, which users often fail to supply. Typos, for example, bring work to a standstill, she says. "They didn't put a space and the script won't run; they put two dashes and the script won't run."

Researchers have to be patient enough to learn the 'grammar' of a programming language — to know how computer commands are structured — before attempting anything grander, says Fournier. "You need to learn how to talk to R so you can do some basic things before you get to the cool modelling." For example, scientists need to be comfortable opening and creating data files or creating subsets of data before starting to run analyses.

Novices often try to learn too many tools at once, says Broman. "They learn a bit of Python and a bit of R or a bit of Ruby rather than concentrating on learning one new thing," he says. He recommends that people pick a language that's popular with their colleagues and work initially in four-hour blocks, which he says provide enough time to work through hurdles and get a sense of progress.

The trick is to stay motivated until the fun of doing real scientific analysis can start. When baseball enthusiast Pat Schloss, a microbiologist at the University of Michigan in Ann Arbor, was teaching himself R, he worked his way through a book of sports analyses called *Baseball Hacks* (O' Reilly, 2006), which shows how to import data and calculate baseball statistics using this language. White tries to make sure that her students are doing actual genetic analyses with R and other tools as soon as possible. "The feeling that they are doing something important and not just busy work helps them retain things," she says. Broman advises trainees to set themselves concrete goals — for instance, making a graph or ordering data — and to try to get a lab friend to learn at the same time (see 'Get with the program').

Resources such as Codeacademy, rOpenSci and DataCamp are plentiful for scientists who want to learn programming on their own. Coursera offers introductory courses such as the Data Scientist's Toolbox. Books such as *Practical Computing for Biologists* (Sinauer, 2011) and many others can also be useful.

ON THE RIGHT COURSE

Broman has posted a series of mini tutorials (see go.nature.com/2jchzrv) on basic topics such as organizing data and accessing R packages. But he thinks that a lot of students benefit from formal classes that provide clear exercises and personal interactions; these, he says, help students get past early stumbling blocks such as getting to the command line or working out why a computer does not respond to a command.

That does not necessarily mean enrolling in a course for computer scientists. Most scientific researchers want to learn code so that they can answer their own questions, not so that they can become fully fledged programmers. They are often most interested in the simplest kind of programming, called scripting — creating programs that automatically perform operations, manipulate data files and follow workflows much as users might do manually. They want to know code well enough to put together programs from what other researchers have done, rather than writing a streamlined program from start to finish. Computer programmers, in contrast, are more concerned with engineering software for optimal speed and performance.

Durso took a programming course taught by Ethan White, who has made his course materials freely available (see go.nature.com/2jw5t3j). The classes were aimed squarely at scientists and assumed no previous knowledge of

GET WITH THE PROGRAM

Doing it right

GETTY IMAGES

A buddy system is a great way to share the programming load if you're new to coding.

If you write code, you must test whether it's reliable — and annotate it so that, later on, anyone (including yourself) can easily see what it's doing. Two publications, 'Best practices for scientific computing' (G. Wilson *et al.* Preprint at https://arxiv.org/abs/1210.0530; 2012) and its recent companion, 'Good enough practices for scientific computing' (G. Wilson *et al.* Preprint at https://arxiv.org/abs/1609.00037; 2016), are useful guides, especially for the self-taught. Here are some highlights.

- **Write programs that people can understand** Use copious annotation and consistent, clear file names.
- **Use version control** Tracking changes with revision-control systems such as Git or Mercurial can help identify and fix bugs.
- **Plan reliability checks** Check that code works as intended, and repurpose bugs into test cases — quick checks of whether code performs as expected.
- **Avoid dumb repeats** Each piece of data should have just one authoritative representation — and not be copied into separate locations, which invites uncertainty and use of outdated information; code should be conceived, written and used as modules, not cut-and-pasted into each place it is needed.
- **Use a buddy system** Track problems, get one person to type while another checks, and review code together. M.B.

programming. He finished the semester feeling comfortable writing some code and teaching himself additional programming techniques as needed. "My successes in the class motivated me to work on programming more outside of class," he says.

Katerina Georgiou, a graduate student studying soil chemistry at the University of California, Berkeley, enrolled in an applied statistics course that featured group projects pairing students who specialized in statistics with those who worked in life or physical sciences. The statistics students got a better idea of how to incorporate 'domain knowledge', such as what data mean and how robust they are, and the science students saw how people with rigorous coding and programming experience break problems into programmable chunks. Even if you can pick up the basics on your own, says Georgiou, it is hard to get fluent in programming without working with others. "Being able to read the code of someone with a stronger background than you is very enlightening."

A shorter, two-day option comes from the sibling non-profit organizations Data Carpentry and Software Carpentry, which have instructors all over the world and have trained thousands of researchers. All the lessons are available online, but in-person instruction is more effective, says Jonah Duckles, executive director of Software Carpentry. Sitting alone in front of a computer to learn data-analysis skills can be discouraging because there are so many ways to get stuck.

Software Carpentry helps researchers to use the command line and programming tools to manipulate files, create graphs and track changes. Data Carpentry offers discipline-specific sessions on how to manage data sets and spreadsheets; it assumes little or no programming experience. Last year, the two non-profits together arranged for trained volunteers to offer more than 350 two-day workshops at research departments and institutions worldwide (host institutions cover an administrative fee plus trainers' travel expenses; information about hosting a workshop or finding upcoming ones can be found on the Carpentry websites). These help people get off to a good start and plug into a network through which they can get more support, says Tracy Teal, head of Data Carpentry. "We want to teach the skills and also instil the confidence that they can keep learning more."

"Being able to read the code of someone with a stronger background than you is very enlightening."

MAINTAINING MOMENTUM

Being familiar with computing tools is not the same as incorporating them into the research routine. Perhaps the biggest barrier is insecurity, says Anelda van der Walt, a volunteer with both Data and Software Carpentry and head of Talarify, a company in Cape Town, South Africa, that provides computational training to geneticists. "Many people think, 'I'll just figure it out on my own first. I'm not good enough yet to ask questions'," she says. Instead, they should seek help from others to gain more skills.

People who can offer help need not be on campus. "The thing that was most valuable to me in getting familiar with R was the network of people on Twitter," says Kara Woo, who taught herself to apply the programming language when she was managing data on microbes in frozen lakes at the National Center for Ecological Analysis and Synthesis in Santa Barbara, California. Many of her Twitter contacts who offered advice were ecologists and biologists whom she had already been following for their research. Other resources include subreddits (themed discussion groups hosted by the social-news site Reddit), the #rstats and #python tags on Twitter and the online community Stack Overflow.

Novices users of these forums should follow certain norms, especially when accessing online platforms such as Stack Overflow that are frequented by hardcore programmers. First, they should search for answers to questions that are similar to their own and that have already been explained. Next, they should describe how they have tried to solve the problem on their own. Experienced programmers will quickly spot mistakes that novices overlook, and they can decipher error messages. But they are likely to get annoyed if a question comes across as one from a teenager who is outsourcing homework.

Small meetup.com groups and hackathons, where coders come together to work on common interests, can provide a comfortable setting in which to ask questions and share problems. Next month, research institutions in Australia and elsewhere are hosting digital literacy festivals, collectively called the Research Bazaar, to help to build support communities. In the end, learning to code is much like getting to grips with any other unfamiliar experimental technique, says Marquitta White: researchers have to identify others to help them learn the necessary skills.

Graduate students who can incorporate programming into research will have their pick of postdoc positions and other offers, says Schloss. Such skills — or access to people who have them — are increasingly necessary for the big-data questions that scientists want to pursue. "If they think they have a lot of data now, in ten years we are only going to have more," he says. "If they don't figure it out now, it's just going to get worse." ■

Monya Baker *writes and edits for* Nature *from San Francisco, California.*

TRADE TALK
Research writer

Daniela Hernandez realized during her PhD at Columbia University in New York City that she preferred writing about science to doing it. Now, she's digital science editor at The Wall Street Journal.

Why did you leave research?
Doing science was lonely: you're in the lab at odd hours, with nobody around. I realized that my strengths were in communication.

Describe the transition.
At Columbia, I interned for ABC News. Then, during graduate school, my principal investigator connected me with the Michael J. Fox Foundation for Parkinson's Research in New York City for some science writing. I also did some paid fact-checking at the National Geographic Channel: as a grad student in New York, the cash certainly didn't hurt.

What was your next step?
While I was at grad school I learnt about the media fellowships offered by the American Association for the Advancement of Science. These place science students in newsrooms around the country. I got one at the *Los Angeles Times*. After that I joined the science-communications programme for the University of California, Santa Cruz, hoping to meet possible future employers and colleagues.

How did that go?
It worked out! I did an internship at *Wired* magazine and got hired — it was great fun. After a year, I moved to Kaiser Health News as a reporter. I got hired by *The Wall Street Journal* last February.

Do you like it?
Yes — it's a lot of fun. Everybody is really thoughtful, collaborative and smart.

Any advice for budding science writers?
It's a really enjoyable job, but it can be taxing. And if you freelance, you need to know how to make ends meet. Reading voraciously is important. You need to know what's going on and how other people cover stories.

INTERVIEW BY JACK LEEMING

This interview has been edited for clarity and length.

CORRECTION

The Careers feature 'Code alert' (*Nature* **541,** 563–565; 2017) gave the wrong affiliation for Andrew Durso. He is at Utah State University in Logan.