ILLUSTRATION BY THE PROJECT TWINS

WHY SCIENTISTS ARE TURNING TO RUST

Despite having a steep learning curve, the programming language offers speed and safety. **By Jeffrey M. Perkel**

In 2015, bioinformatician Johannes Köster was what he called "kind of a full-time Python guy". He had already written one popular tool — the workflow manager Snakemake — in the programming language. Now he was contemplating a project that required a level of computational performance that Python simply couldn't deliver. So he began casting about for something new.

Köster, now at the University of Duisburg-Essen in Germany, was looking for a language that offered the "expressiveness" of Python but the speed of languages such as C and C++. In other words, "a high-performance language that is still, let's say, ergonomic to use", he explains. What he found was Rust.

First created in 2006 by Graydon Hoare as a side project while working at browser-developer Mozilla, headquartered in Mountain View, California, Rust blends the performance of languages such as C++ with friendlier syntax, a focus on code safety and a well-engineered set of tools that simplify development. Portions of Mozilla's Firefox browser are written in Rust, and developers at Microsoft are reportedly using it to recode parts of the Windows operating system. The annual Stack Overflow Developer Survey, which this year polled nearly 65,000 programmers, has ranked Rust as the "most loved" programming language for 5 years running. The code-sharing site GitHub says Rust was the second-fastest-growing language on the platform in 2019, up 235% from the previous year.

> **"The tooling and infrastructure around Rust is really phenomenal."**

Scientists, too, are turning to Rust. Köster, for instance, used it to create an application, called Varlociraptor, that compares millions of sequence reads against billions of genetic bases to identify genomic variants. "This is huge data," he says. "So that needs to be as fast as possible." But that power comes at a cost: the Rust learning curve is steep.

"It does take some up-front time," says Carol Nichols, a member of the Rust core team and founder of the consultancy firm Integer 32 in Pittsburgh, Pennsylvania. "But it has enabled me to do things that I wouldn't otherwise be able to do. I see that time as well spent."

Caution: no guide rails

Workflows for analysing scientific data tend to use languages such as Python, R and Matlab. These interpret lines of code one by one and then execute them, a style of programming that is good for exploring data, but not at speed.

C and C++ are fast, but they have "no guide rails", says Ashley Hauck, a Rust programmer (or 'Rustacean', as community members are known) in Stockholm. For instance, there are no controls that stop a C or C++ programmer from inappropriately accessing memory that

LET'S GET OXIDIZING

Here's how to create a GenBank file reader so you can explore some of the features of Rust.

- Install Rust at www.rust-lang.org/learn/get-started
- Clone the GitHub repository at https://github.com/jperkel/gb_read
- Execute 'cargo run' from the command line to download external dependencies and build the application. By default, the application parses the GenBank file 'nc_005816.gb' in the GitHub repository, but you can specify an alternative input file with 'cargo run <filename>'
- Execute the included tests using 'cargo test'.
- Create and view documentation with 'cargo doc --open'.

has already been released back to the operating system, or to prevent the program from releasing the same piece twice. In the best-case scenario, this would cause the program to crash. But it can also return meaningless data or expose security vulnerabilities. According to researchers at Microsoft, 70% of the security bugs that the company fixes each year relate to memory safety.

Memory rules

Rust's model uses rules to assign each piece of memory to a single owner and enforce who can access it. Code that violates those rules never gets the chance to crash — it won't compile. "They have a memory-management system that is based on this concept of lifetimes that lets the compiler track at compile-time when memory is allocated, when it's freed, who owns it, who can access it," explains Rob Patro, a computational biologist at the University of Maryland, College Park. "There's an entire large class of correctness errors that go away simply by virtue of the way the language is designed."

The same guarantees help to ensure that parallelized code — software written to run on multiple processors — can run safely, for instance by eliminating the possibility that multiple computational threads will access the same data at the same time.

The result is a language that is easier to maintain and debug, but harder to learn. "No other mainstream languages really have these concepts, and they're really core to understanding a lot of how you have to write code in Rust," Nichols says. Stephan Hügel, who studies the visualization of geographical data at Trinity College Dublin, estimates that he spent two or three months porting a Python algorithm for converting geospatial coordinates from one reference system into another into Rust, achieving fourfold faster execution. Richard Apodaca, founder of the cheminformatic-software company Metamolecular in La Jolla, California, says it took him about six months to become proficient in the language.

Focus on usability

To compensate, Rust's developers have optimized the user experience, says Manish Goregaokar, who leads the Rust developer-tooling team and is based in Berkeley, California. For instance, the compiler produces particularly informative error messages, even highlighting offending code and suggesting how to fix it. "If your language is going to introduce a novel concept, it had better be pleasant to work with," Goregaokar explains.

The Rust community also provides extensive documentation and online help, including a popular online reference called the Book and a 'Cookbook' of recipes for solving common problems. Users praise the Rust toolchain — the applications that programmers use to turn code into applications (see 'Let's get oxidizing'). "The tooling and infrastructure around Rust is really phenomenal," Patro says. Unlike the many compilers and ancillary utilities that programmers use to build C code, Rustaceans can use a single tool, called Cargo, to compile Rust code, run tests, auto-generate documentation, upload a package to a repository and more. It also downloads and installs third-party packages automatically. A Cargo plug-in called Clippy flags common errors and 'non-idiomatic' Rust code, a feature that Patro calls "absolutely phenomenal".

There are Rust plug-ins for popular development environments, such as Microsoft's Visual Studio Code and JetBrains' IntelliJ, as well as a Rust 'playground' that provides a live, online Rust environment for code experimentation. And David Lattimore, a software developer in Sydney, Australia, created a 'kernel' for using Rust in Jupyter computational notebooks, as well as a Python-style interactive environment called a REPL (read-evaluate-print loop).

Aiding development is Rust's ecosystem of third-party packages, or 'crates', currently numbering nearly 50,000 (see 'Rust rising'). These encapsulate algorithms in disciplines such as bioinformatics (Köster's Rust-Bio), geosciences (the Geo-Rust project) and mathematics (nalgebra). Still, says Nichols, "that could definitely tip the balance away from Rust, if the libraries you need are just not in Rust". Programmers can sometimes bridge that gap using Rust's 'foreign function interface', however.

Oxidized code

Coding logistics aside, what's undeniable is that Rust is fast. In May, bioinformatician Heng Li at the Dana-Farber Cancer Institute in Boston, Massachusetts, tested multiple languages on a computational-biology task that involved parsing 5.7 million sequence records. Rust edged out C to take the top spot. "When we want to write a high-performance program using multiple threads, and also if you need it to be very fast and also compact in memory, then Rust is the ideal choice," Li says.

Luiz Irber, a bioinformatician at the University of California, Davis, used Rust to recode (or 'oxidize', in Rust parlance) a tool called Sourmash — which performs genomic searches and taxonomic profiling — to ease software maintenance, gain access to modern language features and make the code work in a web browser, he says.

Led by graduate student Hirak Sarkar, Patro's team used Rust to build a gene-expression analysis tool called Terminus after team member Avi Srivastava returned from an internship at 10x Genomics, a biotechnology company in Pleasanton, California, that uses Rust to develop open-source tools. "The beauty of Rust is, it makes the task of debugging very easy, because memory management is much, much better," explains Srivastava, who is now at the New York Genome Center.

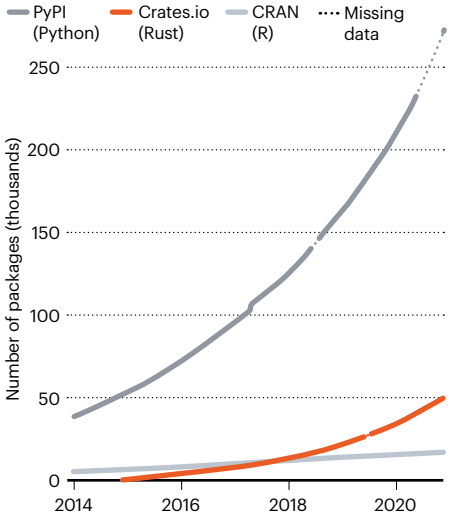
But for many Rustaceans, the human element is equally compelling. Hauck, a member of the LGBT+ community, says that Rust users have gone out of their way to make her feel welcome. The community, she says, has "always made an effort to be extremely inclusive — like, very much aware of how diversity impacts things; very aware of how to write a code of conduct and enforce that code of conduct".

"That's probably a majority of the reason I'm still writing Rust," Hauck says. "It's because the community is so fantastic."

Jeffrey M. Perkel is technology editor at *Nature*.

RUST RISING

The Rust packages repository https://crates.io has grown sharply since 2016, mirroring the rapid uptake of the language.



SOURCE: HTTP://WWW.MODULECOUNTS.COM

Correction

This Technology feature gave the wrong source for the graphic entitled 'Rust rising'. The data came from http://www.module-counts.com, not from https://crates.io. Also, the story erred in stating that the Clippy plug-in is a third-party component. Finally, it erroneously stated that 70% of the bugs that Microsoft fixed each year relate to memory safety. In fact, it is 70% of the security bugs, not all bugs.