Why we created Julia

In short, because we are greedy.

We are power Matlab users. Some of us are Lisp hackers. Some are Pythonistas, others Rubyists, still others Perl hackers. There are those of us who used Mathematica before we could grow facial hair. There are those who still can't grow facial hair. We've generated more R plots than any sane person should. C is our desert island programming language.

We love all of these languages; they are wonderful and powerful. For the work we do — scientific computing, machine learning, data mining, large-scale linear algebra, distributed and parallel computing — each one is perfect for some aspects of the work and terrible for others. Each one is a trade-off.

We are greedy: we want more.

We want a language that's open source, with a liberal license. We want the speed of C with the dynamism of Ruby. We want a language that's homoiconic, with true macros like Lisp, but with obvious, familiar mathematical notation like Matlab. We want something as usable for general programming as Python, as easy for statistics as R, as natural for string processing as Perl, as powerful for linear algebra as Matlab, as good at gluing programs together as the shell. Something that is dirt simple to learn, yet keeps the most serious hackers happy. We want it interactive and we want it compiled.

(Did we mention it should be as fast as C?)

While we're being demanding, we want something that provides the distributed power of Hadoop — without the kilobytes of boilerplate Java and XML; without being forced to sift through gigabytes of log files on hundreds of machines to find our bugs. We want the power without the layers of impenetrable complexity. We want to write simple scalar loops that compile down to tight machine code using just the registers on a single CPU. We want to write `A*B` and launch a thousand computations on a thousand machines, calculating a vast matrix product together.

We never want to mention types when we don't feel like it. But when we need polymorphic functions, we want to use generic programming to write an algorithm just once and apply it to an infinite lattice of types; we want to use multiple dispatch to efficiently pick the best method for all of a function's arguments, from dozens of method definitions, providing common functionality

across drastically different types. Despite all this power, we want the language to be simple and clean.

All this doesn't seem like too much to ask for, does it?

Even though we recognize that we are inexcusably greedy, we still want to have it all. About two and a half years ago, we set out to create the language of our greed. It's not complete, but it's time for a 1.0 release — the language we've created is calledJulia. It already delivers on 90% of our ungracious demands, and now it needs the ungracious demands of others to shape it further. So, if you are also a greedy, unreasonable, demanding programmer, we want you to give it a try.