When Should I Use C?

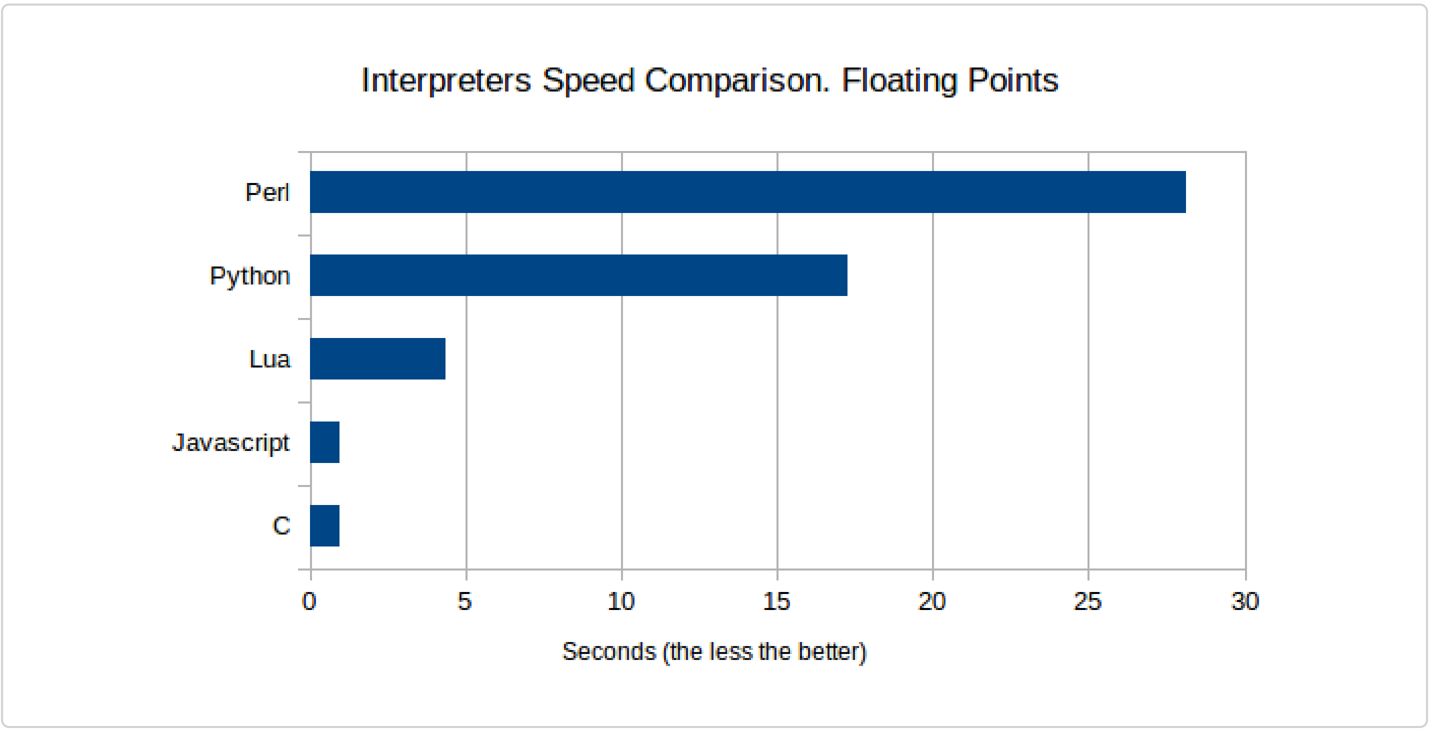
This lesson will show how can we use the optimizations that C offers compared to other languages to our advantage.

My take on scientific programming is that I think of C as one of many tools in my toolkit for performing computational tasks in my scientific work. I wouldn't necessarily suggest *only* programming in C. On the other hand, I would recommend taking advantage of C when the situation calls for it. In our lab, we use Python, R, (sometimes Matlab but increasingly less often), and when we feel the need, the need for speed, we use C.

For interactive data exploration, like when you want to load in some data, plot it in different ways, do some rudimentary calculations, plot the results, etc., then C may not be the best choice. For this sort of interactive exploratory scripting, a language like Python, Matlab, R, etc., may be entirely sufficient. In particular, these other languages make it very easy to generate great-looking graphics quickly.

For cases where you need to process a large amount of data, you will find that these languages are slow. Even for fairly common statistical procedures like bootstrapping (techniques that involve resampling thousands or tens of thousands of times), interpreted languages will be orders of magnitude slower than C.

This is the situation when C starts to become very attractive. If you have a data processing operation or a simulation, and you know it will take a long time to run, then it is often worth it to spend some time implementing it in C. The graph below compares the speed of interpreters for several languages. As you can see, C shines in this regard.



My rule of thumb is that if I have to wait more than about 10 seconds to see the result of a calculation or operation, then I get annoyed, and I think about implementing it in C.

You might think, who cares if my calculation takes 10 seconds, or 30 seconds, not 5 minutes, for that matter? Are 5 minutes so bad? The answer is, no, it's not so bad if you only have to do it once… but it's almost **never** the case that you perform a computation on your data only once.

An Example

Imagine you write some Matlab code to read in data from one subject, process that data, and write the result to a file, and that operation takes **60 seconds**. Is that so bad? Not if you only have to run it once. Now let's imagine you have 15 subjects in your group. Now 60 seconds is **15 minutes**. Now let's say you have four group with 15 subjects in each group. In that case, 15 minutes is **one hour**. You run your program, have lunch, and come back an hour later, and you find there was an error. You fix the error and re-run… another hour. Even if you get it right, now imagine your supervisor asks you to re-run the analysis five different ways, varying some parameter of the analysis (maybe filtering the data at a different frequency, for example). Now you need **5 hours** to see the result. It doesn't take a massive amount of data to run into this sort of situation.

If you program your data processing pipeline in C, and you achieve a 100x speedup (not unusual), now those 5 hours turn into **180 seconds** (you could run your

analysis twice, and it would still take less time).

We’ve learned a lot about the behavior and efficiency of C. We’ll conclude this discussion in the next lesson and move on to writing code.