Technical Computing for the Earth Sciences, Lecture9: Profiling Optimization

EARS 80.03

Optimization

Simplest: just time how long functions take, and # of allocations

- Time how long functions take, and # of allocations with @time
- `@btime` and `@benchmark` from BenchmarkTools are more sophisticated version of this (multiple trials, exclude compilation time and global variable overhead).

Particularly important for Julia: ***type instability***

- Causes excess allocations, lots of other problems
- Check for this with `@code_warntype`. If you see a red `Any`, that's really bad, and if you see a yellow `Union{…}`, that can be a warning sign.

Profiling

Profiling helps you find which parts of your code are taking up the most time. There are several ways of doing this in Julia:

- `Profile` stdlib (ref)
- Juno `@profiler` (ref)
- VSCode `@profview` (ref)

This tends to be most useful for somewhat larger projects where you might not be able to optimize everything by hand / by inspection, and may not even know *which parts* of your 1000s of lines of code matter, or need the most work.