Lesson 4, Week 4: Looking back, looking forward

First of all, congratulations!

You've made it this far, well done! Programming is hard work[1] and if you've got here and done most of the exercises and assignments, you deserve plenty of plaudits!

The looking back: what we covered, how and why. And what we didn't

We covered quite a lot of the basics of Julia: values[2], types[3], variables, functions[4], local and global scope, logical tests, structures of branching code[5].

We applied it all to text we created ourselves and also to one sample text file[6], even learning a bit of Monte Carlo simulation doing so. It's been quite a journey and one could do very substantial projects without using anything beyond the matter of this course.

But in truth, there is much, much more to Julia than that. It is, after all, a very recently created language, building on all that has gone before and aiming providing a very wide audience with ease of writing code, efficiency n extending code, and the possibility of blindingly fast code.

Here's an indication, roughly in order of how accessible the topics will be to you right after finishing this course[7]: improving your coding style; accessing and using Julia packages; making plots and other graphics with Julia; learning how to collaborate with others[8]; user-defined types; Julia's system of modules; high-speed performance; meta-programming . . . there is still much more than that.

But really, don't be intimidated, you need not take all of that or indeed any of that on. You've already come a significant way, and maybe you have gained all you ever want from learning to program. It's fine to pat yourself on the back and do something else[9].

[1] Hard for almost everyone—that's why successful programmers earn good money.
[2] Characters, strings, numbers, arrays, and `true` and `false`
[3] `String, Char, Int64, Float64, Bool` and a few `Array` types also
[4] Including operators. User-defined in three ways: the `function` keyword, inline functions, and anonymous functions.
[5] with `if`, `while` and `for`
[6] Go on, you can do similar on a text file you find or make yourself!
[7] Assuming that this is your only experience of programming so far
[8] The Julia community is very pleasant, polite and helpful; GitHub is a particularly good platform for collaboration
[9] The branching of paths is not limited to code!

If you want to continue with Julia ...

There are many online tutorials and courses. You will gain a lot from read the Julia documentation at `https://docs.julialang.org/`. A very good way to learn more and enjoy yourself at the same time is to do a project that interests you. Finally, you can easily get help online via `http://discourse.julialang.org/` and other resources, for example Stackoverflow at `http://stackoverflow.com/`.

Of course, do search through the packages available for Julia at `https://pkg.julialang.org/`—you'll find plenty to explore and enjoy, and many have tutorials that you will find accessible. You might even find one you like so much that you want to help develop it[10].

I would strongly recommend that you learn to use at least one of IJulia and Juno. They are alternatives to the way we did things on this course: the REPL + `.jl` files.

IJulia is a very clean interface, a so-called notebook, and it allows you to combine detailed discussions with elegant bits of code. Juan Klopper and I use it for the course "Scientific Programming in Julia" which you will find on Coursera, reach them via `http://coursera.org/`.

Juno is a so-called IDE, an integrated development interface, all on one screen you get a REPL, an editor, graphics if you need it, your file system and even other filesystems (particularly on `github.com`) and more. It takes a while to learn how to use it well, though when you do it can really push up your productivity.

What about options other than Julia?

Much as we love Julia, we recognise there many factors that influence choice of language for continuing your adventure with programming. Your workplace may demand a different language, you may have a specialised purpose that is better served by another language, you may feel that one of the better-known languages would be better for your job prospects. And of course, you may simply be curious to see what other languages are like and see whether you like any of them better.

Goodbye and good luck!

[10] It's not that hard, start the online docs. Package developers appreciate help with documentation especially, and as we read we all spot ways that the documentation could improve. It isn't too hard to learn how to make pull requests that eventually are incorporated in the online material.