University Collaboration Budget Final report

Stephen J Eglen

October 11, 2021

1 Summary of work

We thank CUP for supporting us financially on our project to develop resources for mathematics students to learn the Julia language. We have developed a set of freely available materials at: `https://sje30.github.io/catam-julia`, thus meeting our core objective of the application.

Our project took some time to get started initially using a group of Phd students to create material. This proved to be hard as everyone was busy with their research. Once the pandemic struck, we needed to pause the project anyway. We then decided a new, and successful, approach which was instead to hire summer students to work on the material full-time. This proved to be very productive.

2 Case study

We have successfully created a set of materials for introducing the Julia Language to mathematics undergraduates, targeted in particular for the CATAM projects that are part of the Mathematics undergraduate tripos at Cambridge: `https://sje30.github.io/catam-julia`.

The funding was used to hire two talented undergraduate students over the summer of 2021. Rather than write a fresh introduction to Julia language, we instead refer students to an excellent (already freely available) course created at MIT by developers of the Julia language: `https://computationalthinking.mit.edu/`. We then highlighted key aspects of the Julia language that we think students should learn. We have developed case studies of popular mathematical examples that should be accessible to 1st year undergraduates of mathematics.

Finally, in discussion with CUP editors, we have selected a book, Data Driven Science and Engineering by Brunton and Kutz `http://databookuw.com`, and converted some of that code from matlab to Julia. In many cases this conversion was straightforward due to the clearly written code provided by the authors. We also wanted to work on another CUP book, Introduction to Computational Genomics (Cristianini and Hahn, 2006). One immediate problem was that the book's website is no longer available, `www.computational-genomics.net`. Further, when the authors directed to us to code on their personal site, `https://computationalgenomics.blogs.bristol.ac.uk/`, we could not use it, as most of the matlab code in turn depended on further matlab toolboxes.

We have also had several discussions with CUP editors about advantages of different computational notebooks (e.g. we chose Pluto here for elegance and simplicity, rather than the more popular Jupyter notebook) and online computational resources for running code.

Our work has been well-received by the directors of the CATAM in the Mathematics faculty at Cambridge, and we hope that future students will use our resources for their computing teaching and research.

3 Student summaries

I asked the two students to describe their experiences working on the project, and specifically whether they would recommend using Julia. The consensus seems to be that they found Julia might be best suited as a second-language to learn, once a user is familiar with computing concepts and can see the advantages of Julia (speed, single-language design, flexibility). Matlab and python may therefore still (currently) be better choices for students who are new to programming.

Student two has also provided specific feedback on converting the matlab code provided by Brunton and Katz.

3.1 Student one feedback

Julia is a language which I find is particularly slick and satisfying to use. It might just be attuned to the way that I think, but it tends to be easier to translate ideas into code than just about anything else that I have used before. For any mathematical programming that I do in the near future, I expect Julia would be my best option. Perhaps one of my favourite aspects of Julia is that so much of the language and almost all of its packages are written in Julia and are opensource, which can be very useful for debugging what would seem in other languages to be bizarre and unfixable issues.

For CATAM, I think MATLAB remains slightly easier to use for a complete beginner to programming, as Julia is a little less "plug and play". Also I think MATLAB code would remain more readable to a beginner (Julian peculiarities such as short-circuiting if statements and automatic returning of the final statement spring to mind as examples of things unintuitive to a new programmer). Its active development may also be a deterrent to some, as while the base language has been around long enough to be stable, the same might not be true for some of the packages that might be required.

However, Julia would be my first preference if I were to do CATAM-like projects in the future, and I would recommend it to anyone already familiar with programming for similar mathematical projects. If the Faculty so wish, I think that dual support of both MATLAB and Julia would be good to give the students more of a choice. At the very least, I think that the Faculty could do a better job of encouraging students to choose their language carefully, and not just default to MATLAB when it may not be the best tool for the job.

Since they have been rather central to this project, I would also like to comment on Pluto notebooks. I hadn't really used notebooks like this (or Jupyter etc.) before this project, and when we were looking into methods of presentation at the start of the project, they did jump out to me as a great choice but for a couple of minor grievances that for the most part I have managed to work around or solve myself (apart from outputs being above the code, which remains odd). Initially I thought that reactivity was a bit of a gimmick, but the interactivity that it can provide with minimal effort is a great tool to have, even if it doesn't work without opening the file with Pluto (which can take a while). I think that they would actually be an excellent format to present a CATAM project in, were it not for the currently awful PDF conversion.

3.2 Student two feedback

3.2.1 Julia In General:

Before this project I had never used julia before. Learning julia with my preknowledge of mostly c++ and python wasn't a big challenge and it soon became clear to me that in terms of both speed and convenience it is the right language for data science and mathematical computation. With its speed, versatility, convenience and features like broadcasting, macros and many inbuilt functions and tools for all areas of science the language has grown on me and I plan to utilise its full potential going forth.

The biggest drawback of Julia is probably the lack of beginner friendly "out of the box" environments. Also, many of the more specialised julia packages are still experimental and while everything is open source and things like incompatibilities can easily be worked around with some effort, this might scare away people with little previous programming knowledge.

There are a few things that need some getting used to when picking up julia for the first time. Things like different types representing the same thing not being compatible (column and row matrices, vectors, arrays, ranges for example) and the intricacies of type deduction (indices cant be floating point numbers for example).

3.2.2 Thoughts on Pluto Notebooks:

It was the first time that I worked with an interactive notebook. While it is a nice way to create good looking documents combining code, markdown and LaTex, the interface is still very rough and the slow compiling times for larger projects can get quite annoying. Since code is dependent on everything above it, changing small details like plot formatting requires recompiling and rerunning big junks leading to rather long waits during which no work can be done. I do however think there is a lot of potential going forth and these problems could be fixed in the future using things like smaller, parallel scopes or a smarter dependency detection.

3.2.3 Julia for CATAM:

I think that julia is the perfect fit for CATAM. In contrast to MATLAB julia offers so much more in terms of specialisation, active community and future potential. The University should in my opinion do more to encourage a conscious choice of programming language to use for CATAM, offering advise, guidance and examples not just exclusively for MATLAB. I also think that the CATAM projects should be phrased in a more general setting, moving away from the MATLAB centred format they tend to be in at the moment, considering that a good choice of programming language for a given project is already an important skill to work on.

While it is definitely slightly trickier to pick up as a beginner, I think that with resources like the introduction and case studies developed in this summer project it is very feasible to include julia as a second standout alongside MATLAB for the CATAM projects. Julia code can be made to look very similar to MATLAB code and more advanced features like short circuiting, macros and classes can easily be avoided and left for the more advanced students to play around with.

3.2.4 Julia and the book "Data-Driven Science and Engineering":

When translating MATLAB code from the book into julia code, a few things stuck out to me: For the most part the code translates very easily with the only main difference in core functionality being the use of square brackets for indexing in julia (which I like a lot more as it differentiates indexing and function calls). I ran into a few bugs caused by the difference in type deduction between the languages, mainly due to the fact that in MATLAB everything defaults to floating point numbers while in julia types are more complicated than that. Another problem I ran into occasionally was differing conventions with things like function parameter order or matrix orientation (for example ranges in MATLAB are treated as row-arrays while in julia they are column-vectors).

For loading the data files julia has a nice package for dealing with MATLAB files, however it would have been nice to have data stored in a format independent to the programming language used (CSV or HDF5 etc). I also found it somewhat confusing that functions are used across files without any mention of the place they are defined, a MATLAB feature I very much disagree with from a design point of view. I also found that for many MATLAB files the largest part of the code consists of graph formatting, something julia solves much more elegantly than CATAM in my opinion.