JuliaEpiHandbook

A Guide to Julia for Epidemiologists

Callum Arnold

5/20/23

Table of contents

About This Project

The purpose of this book is to create a similar document as the excellent EpiRHandbook for epidemiologist who are interested in becoming Julia language users. Most people in epidemiology and statistics use R, Stata, or SAS. There are still plenty of people who exclusively use Excel to great results. That's OK, and I'm not trying to tell you that those options are bad and you should convert to Julia immediately. But there are some advantages that I think make it an attractive option for epidemiologists to consider.

Motivation

> Why write a book for Julia?
>
> What's wrong with R, or Stata, or Python, or (insert your favourite language here)?

Excellent questions. I use R every day and think it's a great language for epidemiology, statistics, visualization, and many more things. But it's not exactly a fast language. And for me, and plenty more epidemiologists, that can be a problem when working with large data sets or running simulations. Large data is less of an issue because package developers often have already rewritten the core code in a faster language, typically C++. But this is less that ideal if you want to look under the hood as it requires you to learn a second language (and C++ isn't exactly known for being friendly to newcomers). The bigger issues, though, come when you need to write something from scratch, such as creating a large and complex simulation model, R is sometimes just too slow to actually run on your laptop in a reasonable time period, breaking up the development workflow. In this instance, you have two options: 1) suck it up and just hope your first attempt at the code was correct and don't need to change anything, or 2) learn and rewrite your slow code in something faster. This is the classic "2 language problem", and there are plenty of excellent articles on this, such as this Nature Methods article about Julia for Biologists, and this blog post from the creators of Julia.

> OK, but not everyone needs to run big simulations or work with big data. Why learn Julia over R in that case?

Another good question that covers a decent chunk of working epidemiologists. If you don't need performance, I think it's a slightly less compelling case, but there are still plenty of good reasons to think about learning Julia.

Firstly, it has an excellent syntax and design structure that makes it one of the easier (and fun!) languages out there to learn. It's incredibly helpful to be able to think in plain language and pretty much just implement that in code - for loops are quick so you don't need to do mental gymnastics to think about how to *vectorize* your code. I know that's a jargony term that might not mean anything to you, but in slower languages, you have to think about more abstract and unintuitive ways of implementing what you want, whereas much of your instincts *are idiomatic* in Julia i.e. are a fast and preferred implentation. This means that you can do more things more easily in Julia than you could in R, which has shoehorned many capabilities into it with varying degrees of success. Metaprogramming and multiple-dispatch are killer features that are core to how Julia works, and will eventually allow you to write simple and interpretable code that would require tons of `if` statements that result in hard-to-read code in other languages, such as R. And the package manager is excellent, making it so easy to create fully reproducible code, whereas R and Python almost seem to be it intentionally hard to do.

Secondly, and this is a pretty big one, but Julia is a general purpose language whose syntax is rather similar to Python a lot of the time. Whether we like it or not, Python isn't going anywhere. There's a decent chance that at some point you will come across some Python code, or maybe even need to write some for a collaboration, and knowing Julia will make it pretty simple to pick up the key points, without needing to rely on using slow Python in your day-to-day life.

Finally, if you need it, Julia is first-rate when it comes to scientific computing. A lot of aspects of epidemiology rely on differential equations, math, and increasingly, machine learning. Not only is there the excellent SciML ecosystem that houses the largest collection of ODE solvers of any language, but also, because Unicode symbols are valid code in Julia, you can literally write mathematical equations that you lifted out of a paper or book, and there's a decent chance it'll run with only minor modifications!

It Can't All Be Sunshine and Roses

Correct. Julia is by no means perfect, and you will have to decide if the benefits outweigh the tradeoffs (I believe they do for a lot of us)!

Firstly, while Julia has an active, friendly, and very enthusiatic community that is often willing to help when you get stuck, it is a relatively new language so support will be more limited than can be found in others. Related to this, not everything you will want to do will have a perfect pre-built package ready to go. Most of the time, this actually isn't too big a deal as it's either simple enough that you can quickly figure it out, or complex enough that you probably want to have ownership over the implementation, but occassionally it is genuinely an issue. Both of these issues, however, are improving rapidly as the Julia userbase grows and more people share their questions, expertise, and even turning solutions into packages that can be shared!

The next common stumbling block is related to performance. Despite what I said about it being a very fast language, it's quite a bit slower at the start of a session than R or Python, for example. This is known as the "Time to first X" (TTFX) problem, and while everything will be almost instantaneous the second time you run it, it can be annoying. Thankfully, with the recent release of Julia 1.9.0, this is effectively a problem of the past, with most Julia sessions loading packages and creating your first plot within a few seconds, not a minute.

And the last main issue with Julia is that packages aren't always documented as clearly as in other languages, and debugging can be a painful experience. Every package is different, and the Documenter.jl package does an excellent job of standardizing package documentation, but unfortuately not everyone does a great job of writing robust tests and guides to help the user. Sometimes you just need to dig in the weeds to figure something out.

But, that's part of the motivation for writing this book. For a lot of what we do as epidemiologists, we can work with established packages that have solid code bases, so we won't need to worry too much about the correctness of the code. Instead, this book is to help guide someone new to Julia through the core tasks of epidemiology.

Who Is This For/Pre-Requisite Knowledge?

Because this book aims to provide ground-up instruction on using Julia as an epidemiologist, it does not assume any prior coding experience. Many people get started with Excel analysis, and this book tries to provide a new path that will be easier and more reliable for those individuals moving forward. As a result, the earlier chapters will lay out fundamental concepts of Julia and best practices for setting up a project, including concepts on version control and Git. But if you've got experience with R (or another language), then I hope to try and draw parallels and examples that will help speed up your learning.

Keywords, Code, and Other Formatting

Throughout the book, you'll see some keywords, code, and other points that I'll try to delineate with the following formatting:

ℹ Note

This will be a note, and will be used to highlight important points, or to provide additional information.

💡 Tip

This will be used to highlight a useful tip.

⚠ Warning

This will provide a warning that you may get an unexpected result if you're not careful.

- `code` will be used to highlight code.
- `{package.function()}` will be used to denote a specific package and function, e.g., `{DataFrames.subset()}` denotes the `subset()` function from the `{DataFrames}` package. Some languages use a different convention, such as `package::function()` in R, but so where relevant, I will do my best to provide the correct syntax.
- **keywords** will be used to highlight keywords and phrases, e.g., **Git** or **GitHub**.
  - **actions** will also be highlighted in this way, e.g., **commits** or **pushed** being the result of the code `git commit` or `git push`
- ***files*** will be used to highlight file names, e.g., ***README.md*** or ***LICENSE***.
- *italics* will be used for emphasis in certain circumstances, e.g., signifying a question from an interactive terminal command.

Built With

- Quarto
- Julia

Contact

You can contact me via my email: “arnold dot crk at gmail dot com”.

Contributing

If you see any issues, please open an Issue or Discussion on the book's GitHub page. Or, if you know Git, you can open a Pull Request to fix the issue directly, which is even better!

Tip

You can directly go to the code for each page by clicking the "Edit this page on GitHub" link at the bottom of each page's TOC on the right. This can be useful if you would like to point out an issue or propose a change.

If you are interested in contributing text, please email me and we can make it happen!

Acknowledgements

- EpiRHandbook for inspiration

License