

Using the Computational Power of Julia to Drive Clinical Trial Simulations (CTS)

Patrick Hannon,
AD Biostats, MMS Holdings



Julia, The Programming Language

Positive Features

- Multi-paradigm – combining features of imperative, functional, and object-oriented programming
- Syntax is clean and uncluttered
- Multiple dispatch is simply amazing – the ability to define function behavior across many possible combination of argument types
- Dynamic type inference at compile time if you have not explicitly stated in code. You can explicitly define var types, but often it's superior not to do so.
- Macro/meta programming features are 2nd to none (LISP like ability to manipulate code, i.e. code is data)
- Julia developed with HPC in mind, a driving design feature
- Incredibly fast, implementations of code can sometimes outperform versions written and compiled with C++ and FORTRAN



Julia, The Programming Language

Negative Features

- The “Time to Plot” problem – if you’re plotting something simple and want it very quick (from thought/idea to final product), another language/software package is going to be “faster”.
- Package ecosystem is not as well developed as C++, Python, R, etc., ... currently
- Frequent version updates (1.6 or 1.7 may be the next long term stable release version)

Julia’s mandate is to solve the two-language problem, where one uses a high-level (slow) language to develop the code/application (the prototype), and then later convert code to a low-level (fast) language (the production version).



Julia's Remarkable Ascendant Adoption

Many universities, commercial and government research organizations (e.g. Argonne, Los Alamos, DARPA, etc, ...) have aggressively embraced the use of Julia where HPC is involved (typically simulation problems)

Just a few concrete examples of adoption:

FAA Using Julia to develop the Next-Generation Airborne Collision Avoidance System

[Safer Skies - Julia Computing](#)

DAPRA Award to Accelerate Electronics Simulation by 1000x (2021)

<https://juliacomputing.com/media/2021/03/darpa-ditto/>

Celeste Project – cataloging all stars and galaxies in the visible universe (2017)

<https://www.nextplatform.com/2017/11/28/julia-language-delivers-petascale-hpc-performance/>

Julia Programming's Dramatic Rise in HPC and Elsewhere (2020)

<https://www.hpcwire.com/2020/01/14/julia-programmings-dramatic-rise-in-hpc-and-elsewhere/>

Julia Update: Adoption Keeps Climbing; Is It a Python Challenger?

<https://www.hpcwire.com/2021/01/13/julia-update-adoption-keeps-climbing-is-it-a-python-challenger/>



Clinical Trial Simulations (CTS)

- CTS represents an expansive scope and effort of what is being attempted.
- This process generates an extensive number of artificial data points (or sets of data) via probabilistic/stochastic/clinical models (possibly RWD informed).
- These simulated data are used to derive outcomes, possibly model based, then analyzed similar to how the proposed experiment would be.
- Results are assimilated to produce predictive outcomes or general measures of the experiment's operating characteristics.
- The outcome of these simulation results are used to inform how to make trials more efficient (i.e. smaller/shorter/less of them) and shorten the time (and number of subjects) needed to complete the clinical development and obtain regulatory approval with higher probability, and without jeopardizing mischaracterizing the efficacy and safety profile of the drug/device.
- Specific examples: Pharmacological modeling (PUMAS), pharmacokinetic modeling (popPK) for dose selection, enrollment modeling, ***adaptive study operating characteristics modeling***, etc, ...



FDA Updated Guidance for Adaptive Studies (2019)

- Updated guidance describes a process by which a complex highly adaptive study design (with no closed form solutions or explicit guarantees of preserving type I error probability or ensuring adequate power) may be allowed if a comprehensive and compelling simulation is prospectively employed and reported as part of the planning of a study and its protocol development.
- [Adaptive Designs for Clinical Trials of Drugs and Biologics \(fda.gov\)](https://www.fda.gov/oc/adaptive-designs-clinical-trials-drugs-and-biologics)
Section VI.A “Simulations in Adaptive Design Planning”
- A “simulation plan and report” subgroup was initiated by DIA working group “Adaptive Design Scientific Working Group (ADSWG)”
- A good article proposing how to put together a simulation report (content and format)
- [Full article: Simulation Practices for Adaptive Trial Designs in Drug and Device Development \(tandfonline.com\)](https://www.tandfonline.com)
- The lightest of web searches will reveal a multitude of other groups with papers and presentations on this topic.



Preferred Programming Platform for CTS: The Case for Julia, and Challenges

- Julia solves the two-language problem, it is a high-level language with speeds comparable to compiled C++ and FORTRAN;
- Easy to learn syntax lends to clear and transparent understanding of what code is doing;
- Parallel processing is inherent to the language and easy to implement for beginners using high-level array abstractions.
- Julia is Free.
- Julia is built for high-performance and ideal for simulations and intensive calculations.



Preferred Programming Platform for CTS: The Case for Julia, and Challenges

- Package availability may be one of the drawbacks. This is changing quickly.
- Performance may suffer if a particular method/function is not available in the Julia package eco-system, and you have to make external calls to code written in a slow language (e.g. R, Python, etc., ...),
- ***Julia's stat and stochastic methods package eco-system needs to be expansive and complete, and written correctly (proper use of multiple dispatch to help ensure the unreasonable compatibility of package eco-system)***
- Think Big and Go Big: Stat and stochastic method packages should be developed to be deployed under two possible scenarios a) as “linear code” without use of parallel processing, and b) as code that makes use of cluster or GPU acceleration (parallel processing).



Donald Rubin's Prophetic Forward in a Textbook for Multiple Imputation

... Stef is certainly correct when he states in Section 2.1.2: “The idea to create multiple versions must have seemed outrageous at that time (late 1970’s). Drawing imputations from a distribution, instead of estimating the ‘best’ value, was a severe breach with everything that had been done before.” I remember how this idea of multiple imputation was even ridiculed by some more traditional statisticians, sometimes for just being “silly” and sometimes for **being hopelessly inefficient with respect to storage demands and outrageously expensive with respect to computational requirements.**

Some others of us foresaw what was happening to both (a) computational storage (I just acquired a 64 GB flash drive the size of a small finger for under \$60, whereas only a couple of decades ago I paid over \$2500 for a 120 KB hard drive larger than a shoebox weighing about 10 kilos), and (b) computational speed and flexibility. **To develop statistical methods for the future while being bound by computational limitations of the past was clearly inappropriate.** Multiple imputation’s early survival was clearly due to the insight of a younger generation of statisticians, including many colleagues and former students, who realized future possibilities.

- *Flexible Imputation of Missing Data* by Stef van Buuren 2012 First Edition, Second Edition 2018

Donald Rubin's Prophetic Forward in a Textbook for Multiple Imputation

- ... Stef is certainly correct when he states in Section 2.1.2: “The idea to create multiple versions must have seemed outrageous at that time (late 1970’s). Drawing imputations from a distribution, instead of estimating the ‘best’ value, was a severe breach with everything that had been done before.”
- This idea of multiple imputation was even ridiculed by some more traditional statisticians, sometimes for just being “silly” and sometimes for **being hopelessly inefficient with respect to storage demands and outrageously expensive with respect to computational requirements.**
- *Flexible Imputation of Missing Data* by Stef van Buuren 2012 First Edition, Second Edition 2018



Donald Rubin's Prophetic Forward in a Textbook for Multiple Imputation

- **Some others foresaw** what was happening to both (a) computational storage, and (b) computational speed and flexibility.
- **To develop statistical methods for the future while being bound by computational limitations of the past was clearly inappropriate.**
- Multiple imputation's early survival was clearly due to the insight of a younger generation of statisticians, including many colleagues and former students, who realized future possibilities.
- *Flexible Imputation of Missing Data* by Stef van Buuren 2012 First Edition, Second Edition 2018
- Moore's Law states that the number of transistors on a microchip doubles about every two years, though the cost of computers is halved. Another tenet of Moore's Law says that the growth of microprocessors is exponential.
- <https://www.investopedia.com/terms/m/mooreslaw.asp>



PHUSE Julia Initiative for High Performance Computing

- PHUSE Working Group: Data Visualization and Open Source Technology
- Learn how to develop high performance code in Julia to achieve the intended benefits.
- Helpful tips and instructions on how to get Julia and a good editor/debugger installed, links to learning materials, and links to code examples.
- Provide a method(s) of collaboration with groups developing/improving methods and packages (specifically statistical/stochastic/data science).
- Provide a PHUSE repository for collaboration.
- If enough membership emerges, may split out sub-working-groups.
- Come join us at the CSS (Computational Science Symposium) Sep 13th – 15th



Julia, The Programming Language

Good Starting Links:

Good Starting Links:

[The Julia Programming Language \(julialang.org\)](https://julialang.org)

The Julia Language (V1.6.1) – this will likely be the resting place version for Julia for some time into the future (Long Term Support LTS).
Maybe v1.7 might be the next LTS version.

<https://julialang.org/blog/2021/03/julia-1.6-highlights/>

The Unreasonable Effectiveness of Multiple Dispatch

https://www.juliaopt.org/meetings/santiago2019/slides/stefan_karpinski.pdf

JuliaCon2021 (and previous years)

<https://juliacon.org/2021/>