APIs & Community: Building for Success

juliacon 2022

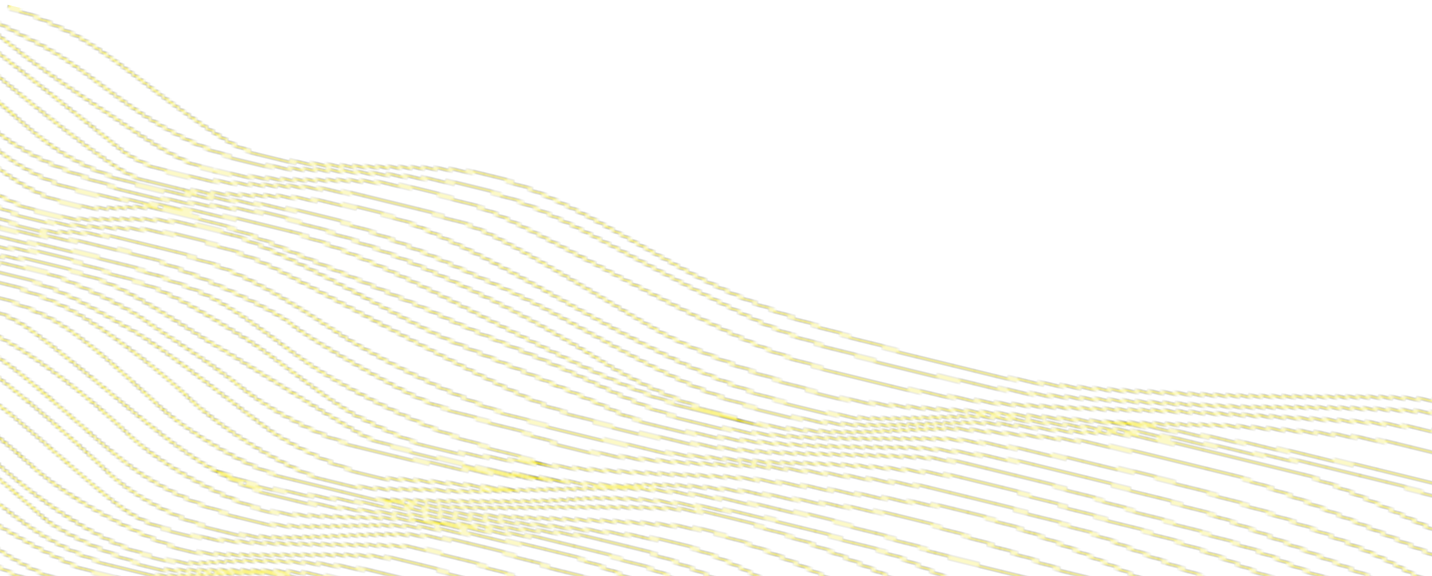
H2O.ai

Erin LeDell Ph.D.
@ledell

Agenda

- Introduction
- R ~~vs~~ and Python (and Julia)
- History of interfaces
- Multi-language software stacks
- APIs > benchmarks
- Community is everything!

Introduction



Hello JuliaCon! 👋

Why am I here? 😅

- First discovered Julia in grad school circa 2013-14 and used it for my research.
- Very involved in the Python and especially R machine learning communities.
- Long history of community organizing!
- Work as Chief Machine Learning Scientist at H2O.ai developing open source ML/AutoML.

Julia: First Impressions

- After working with both C++ and Java for high performance computing, Julia was a dream come true! 🤷‍♀️
- @distributed FTW
- ML ecosystem very immature when I started. (e.g. no AUC function).
- I made heavy use of GLMNET.jl (thanks Simon Kornblith!)

Julia: First Impressions

- During my PhD research, I could not easily do my simulations in R.
- Required training 100,000 ML models to measure correctness of new confidence interval method.

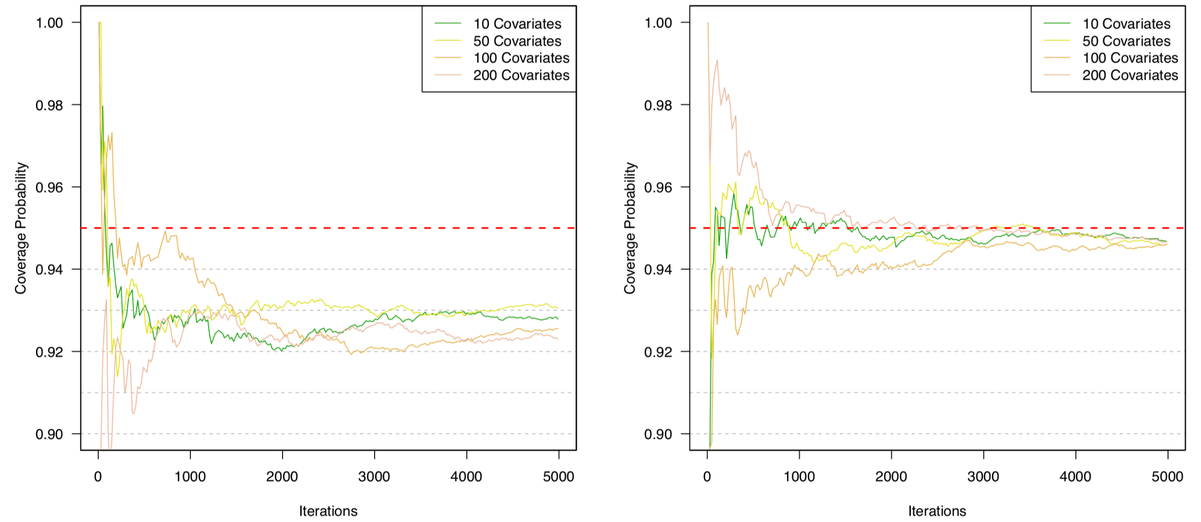
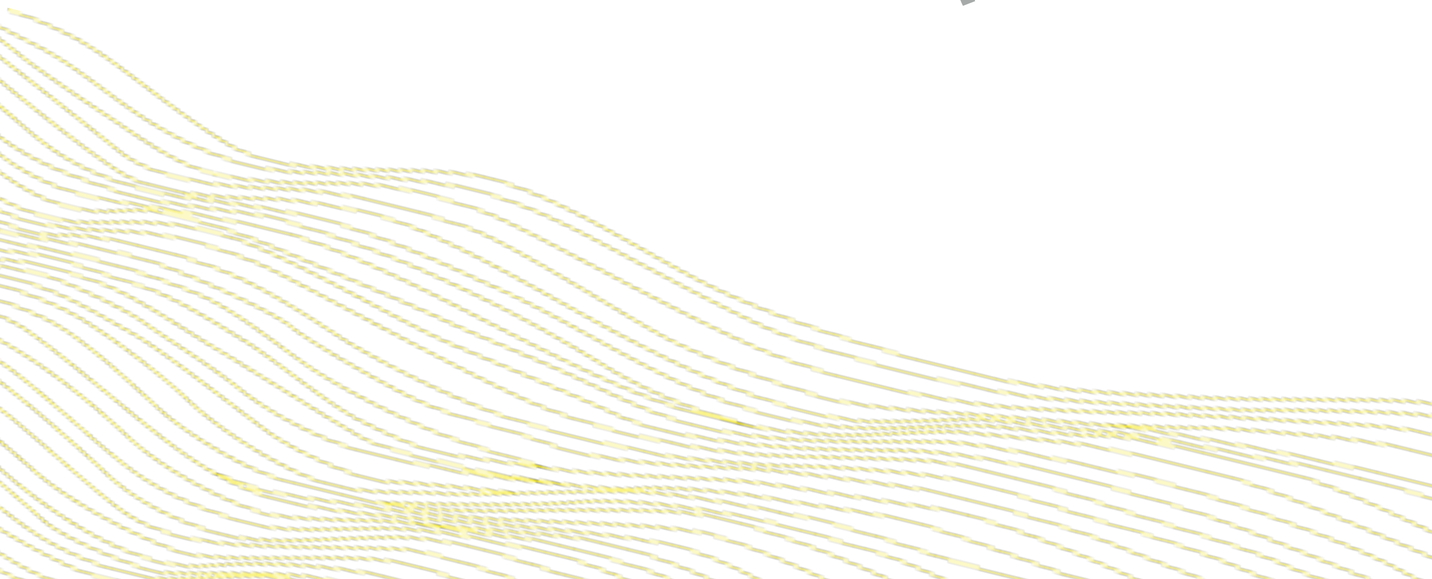


Figure 5.1: Plots of the coverage probabilities for 95% confidence intervals generated by our simulation for training sets of 1,000 (left) and 5,000 (right) observations. In the case of a 95% confidence interval, the coverage probability should be close to 0.95. For the smaller dataset of $n = 1,000$ observations, we see that the coverage is slightly lower (92-93%) than specified, whereas for $n = 5,000$, the coverage is closer to 95%.

Julia is great for running simulations!!!

R ~~vs~~ and Python (and Julia)



R vs Python: The Language Wars

- As data science gained in popularity (2010s), competitive attitudes arose and created the "R vs Python wars".
- Endless blog posts about R vs Python…
- Some were helpful, with the aim of trying to help people new to data science choose a first language to learn.
- Others not so helpful: some Python folks made a habit of "making fun" of the R language. 🙄

R vs Python: The Language Wars

- Language wars narratives usually focus on the "wrong" things.
- *Personal opinion* – What matters most in terms of how happy people are using a language in their day-to-day work is: APIs and Community
  More on that later…

R vs Python: Myths

- ❌ Many incorrect assumptions are made in these posts.
- RStudio has a blog post debunking many of these myths. 👇

Myths about R vs. Python

There are a few common myths that we frequently hear from different organizations struggling with the decision of R vs. Python:

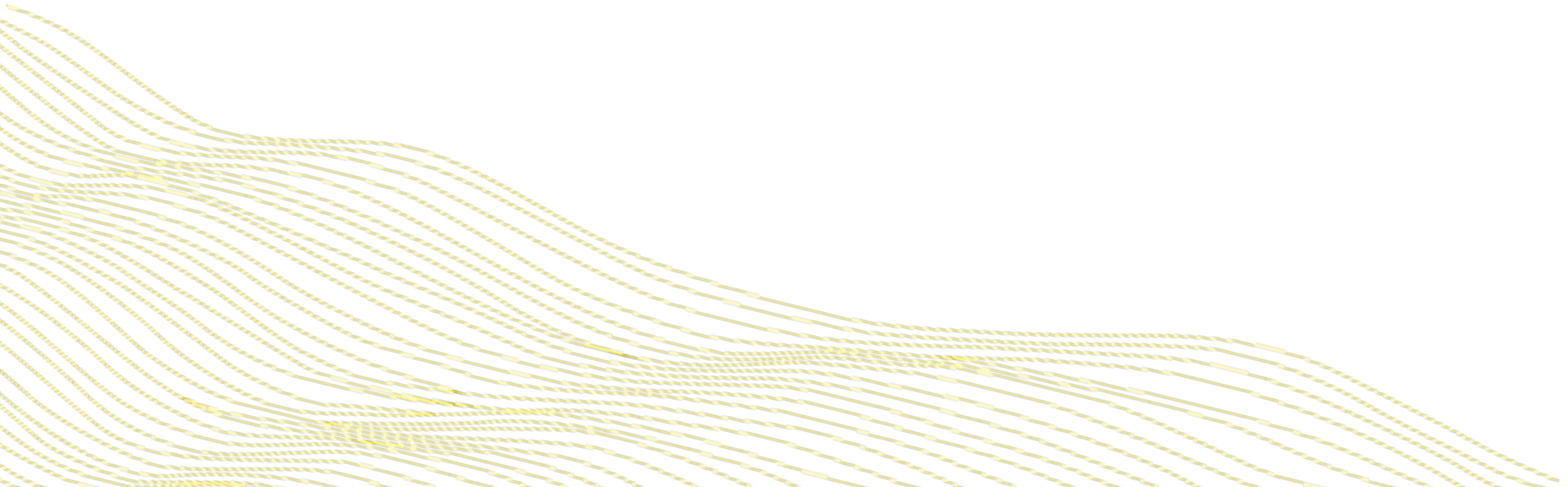
- **Cognitive overload for Data Scientists**: Practitioners often fear that using more than one language will add overhead and context switching, forcing them to use different development environments.
- **Unnecessary burden on IT**: The DevOps and IT teams are concerned that supporting two languages will mean supporting twice the infrastructure for development and deployment, and answering twice as many support tickets for help.
- **Blockers to collaboration, reuse and sharing**: The leaders of data science teams worry that allowing multiple different languages will make it harder for the team to collaborate, re-use each other's work, and deliver that work to the rest of the organization.

https://www.rstudio.com/blog/debunking-the-myths-of-r-vs-python/

R *and* Python

- You do not need to "pick" one language to use. Most experienced data scientists will know at least a bit of both R and Python.
- In the 2020s, it's very easy to use R and Python together in a single data science stack.
- The idea that you must choose one vs the other is an outdated concept!
- R *and* Python *and* Julia...

Interfaces



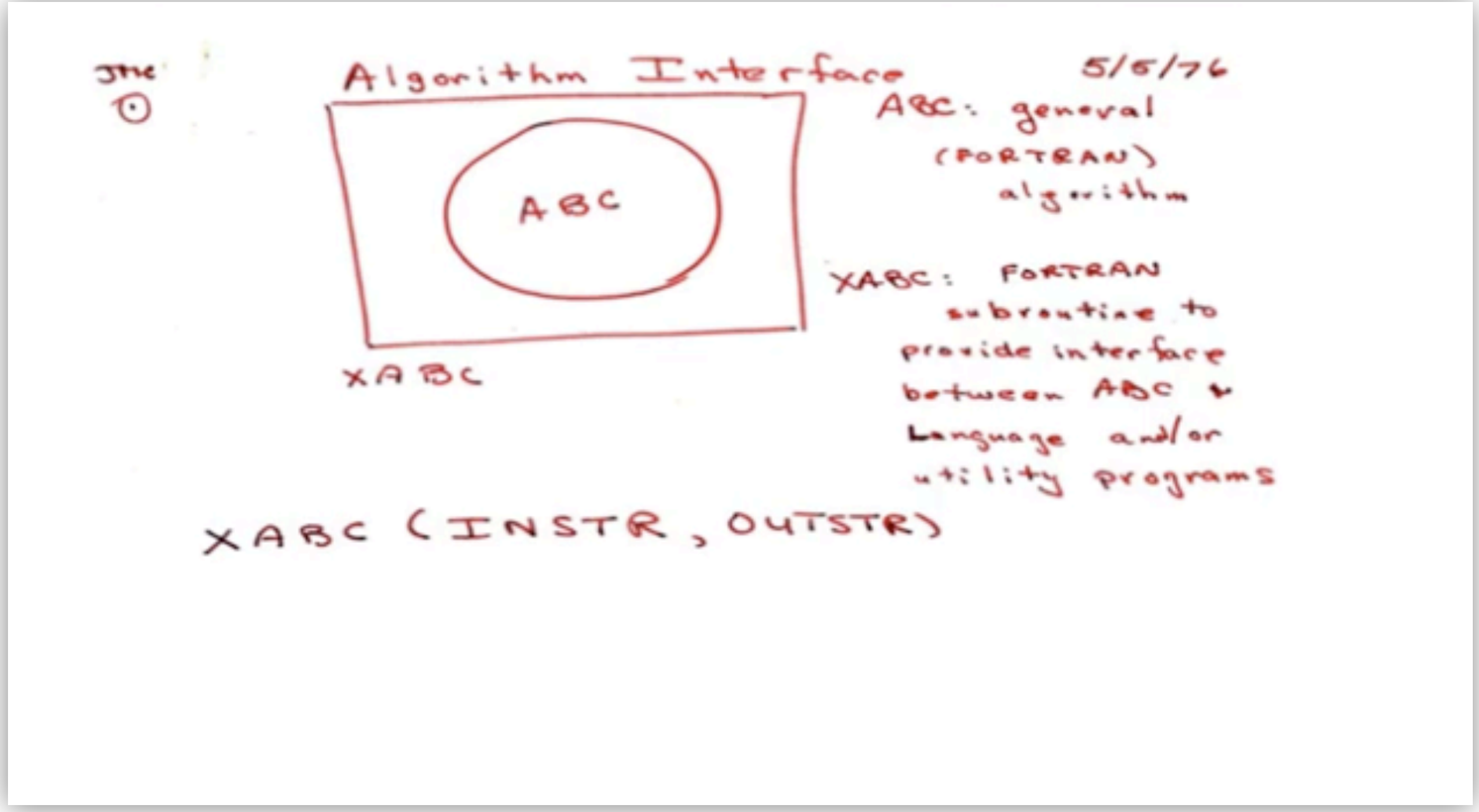
History of Interfaces

John Chambers designed the "Interface Language" designed to interface with FORTRAN.

http://datascience.la/john-chambers-user-2014-keynote/

History of Interfaces

John Chambers (Creator of S language) in 1976: How to make FORTRAN more user friendly?

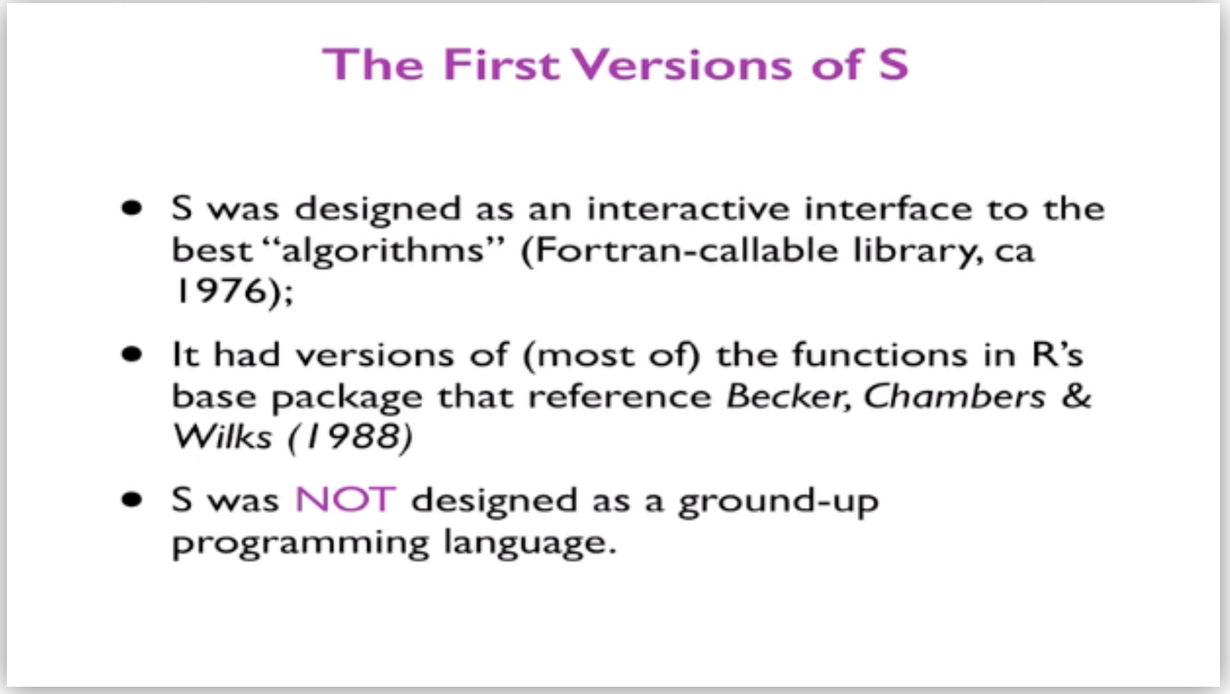


http://datascience.la/john-chambers-user-2014-keynote/

History of R

The S language is the precursor to the R language, an open source reimplementation of S.

The goal was "turn ideas into software, quickly and faithfully."



http://datascience.la/john-chambers-user-2014-keynote/

History of Python

On the other hand, Python was designed by Guido van Rossum to be a general purpose language with the goal of being understandable in plain English.

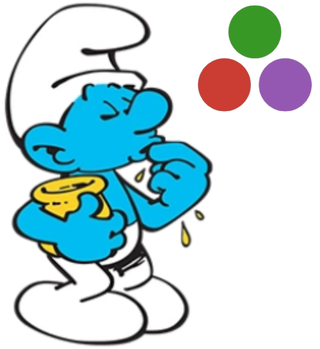
> *"Over six years ago, in December 1989, I was looking for a hobby programming project that would keep me occupied during the week around Christmas. My office would be closed, but I had a home computer, and not much else on my hands. I decided to write an interpreter for the new scripting language I had been thinking about lately: a descendant of ABC that would appeal to Unix/C hackers. I chose Python as a working title for the project, being in a slightly irreverent mood and a big fan of Monty Python's Flying Circus."*

https://www.geeksforgeeks.org/history-of-python/

History of Julia

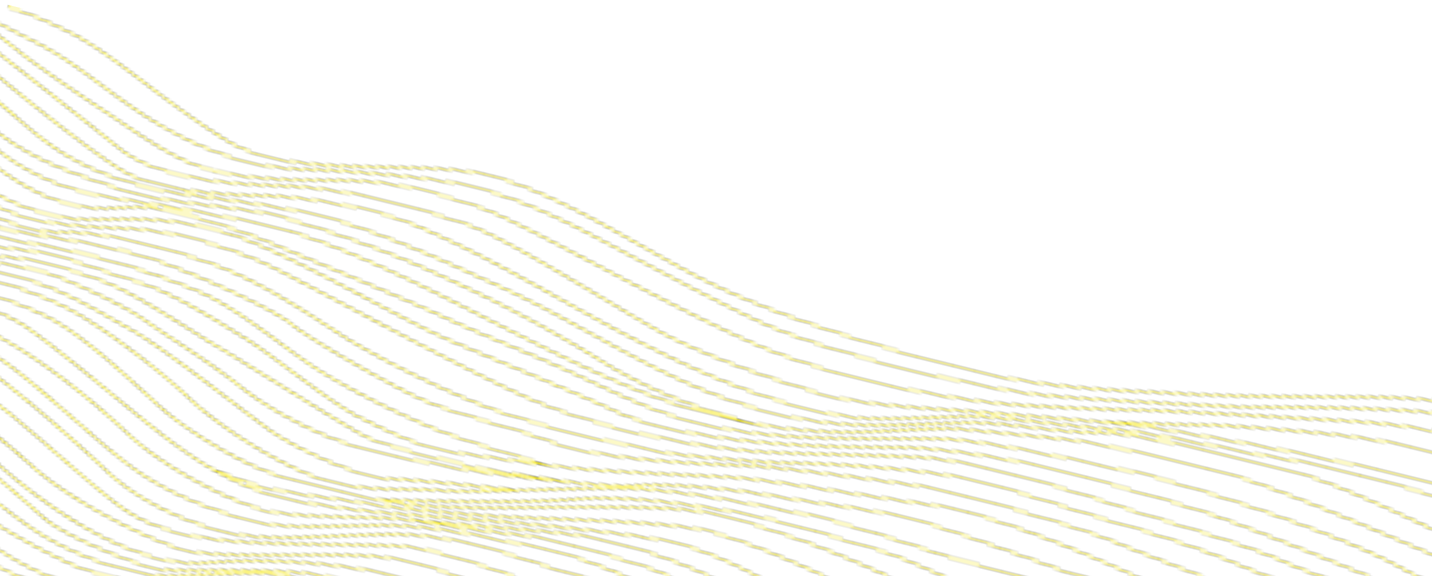
Julia was designed to be as fast as C, with the dynamism of Ruby, interactive and compiled, general purpose, good for stats and math, distributed, etc.

R, Python, C++, etc. all have trade-offs, and Julia was created in order to have it all, because they are “greedy”!



https://julialang.org/blog/2012/02/why-we-created-julia/

Multi-language Software Stacks



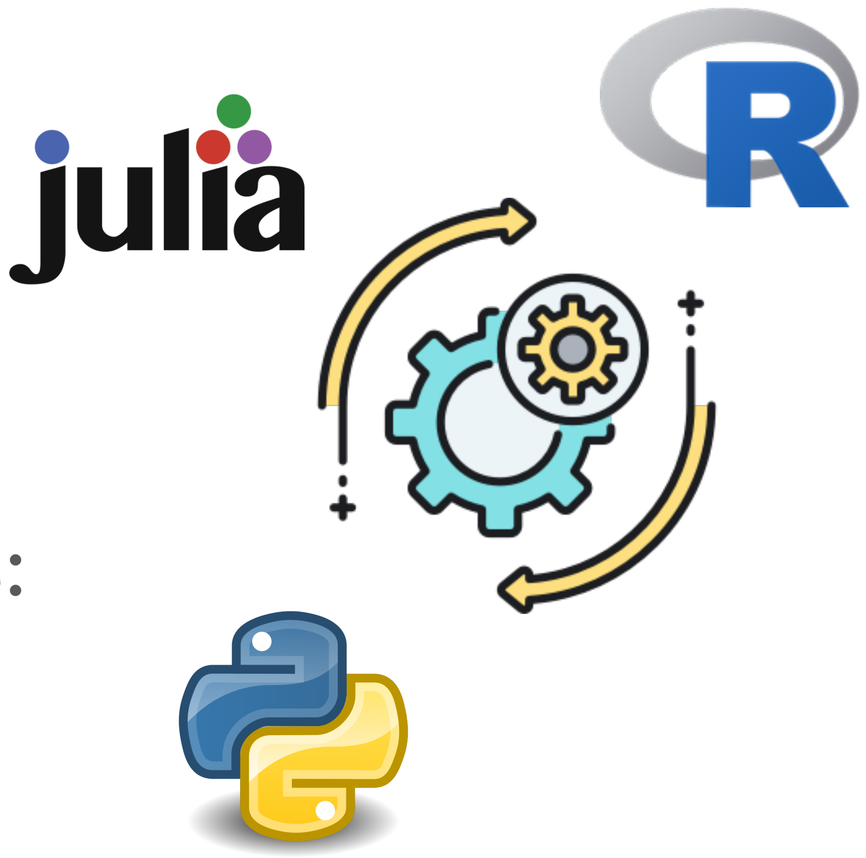
R ↔ Julia ↔ Python

Julia packages:

- RCall.jl
- PyCall.jl

Python packages:

- PyJulia



R packages:

- JuliaCall
- JuliaConnectoR
- XRJulia (John Chambers)

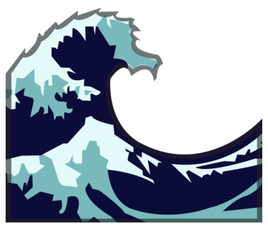
Multi-language APIs

Many popular Machine Learning frameworks are written in one language and expose APIs in other languages.

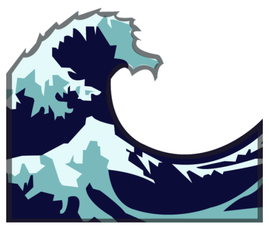
- TensorFlow: C++, Python, R (via RStudio)
- Torch: Lua, Python, R (via RStudio)
- scikit-learn: C++, Python
- H2O: Java, Python, R, Scala

H2O Machine Learning Platform

- Distributed (multi-core + multi-node) implementations of cutting edge ML algorithms and AutoML.
- Core algorithms written in high performance Java.
- APIs available in R, Python, Scala; web GUI.
- Easily deploy models to production as pure Java code.
- Works on Hadoop, Spark, EC2, your laptop, etc.





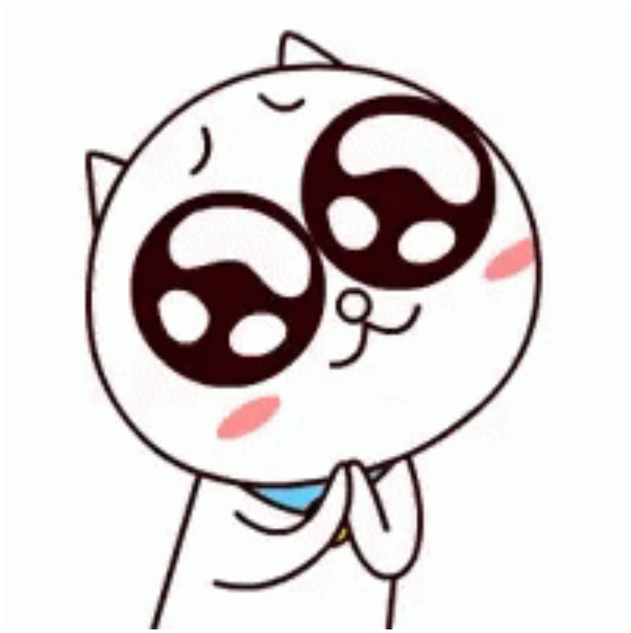
Maintaining Multi-language APIs

- ⛏️ Some tools manually maintain the APIs (e.g. add new function to core, must also write new function to add to Python API).
- ⚙️ H2O's machine learning APIs are largely autogenerated: when a new algorithm or parameter is added to the core, the build process re-generates the R/Py APIs from scratch with the new features. More work up-front, less work long-term.

Communication between API and Core

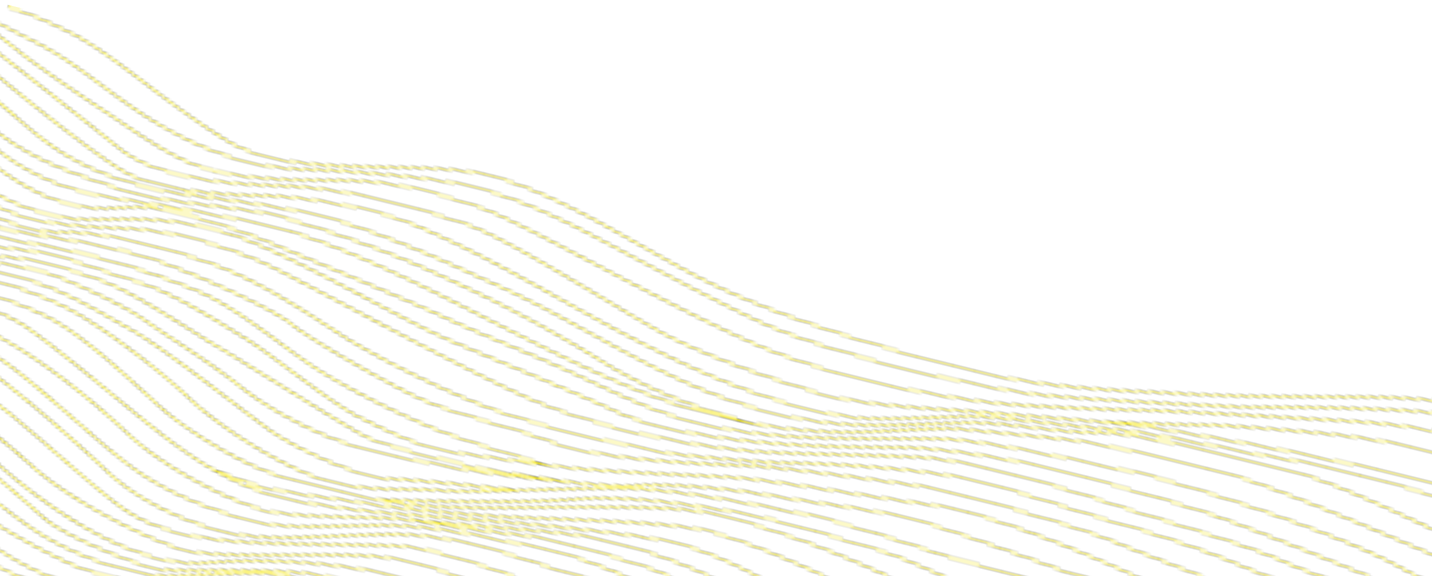
- In H2O, we connect to the Java runtime from R or Python (or any other language) via REST calls. All data is stored in Java memory, no local copies in R or Python. Data is stored in “H2OFrame”.
- Other tools translate between R and Python objects, e.g. R matrix ↔️ NumPy array, converting various data types into local versions.

R and Python with Julia core



It would be wonderful to see more R and Python packages using Julia as the core computation underneath the hood instead of C++ or FORTRAN.

APIs > Benchmarks



How to Evaluate a Language

Benchmarks (usually based on speed) are a primary way that languages get compared.

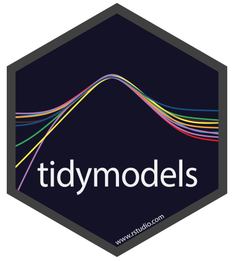
This is useful, but not the whole story!

Many more dimensions:

- Readabilty & extensibility
- Speed from idea to implementation
- Package ecosystem
- Visualization
- Documentation & help (community)

How to Evaluate a Machine Learning Library

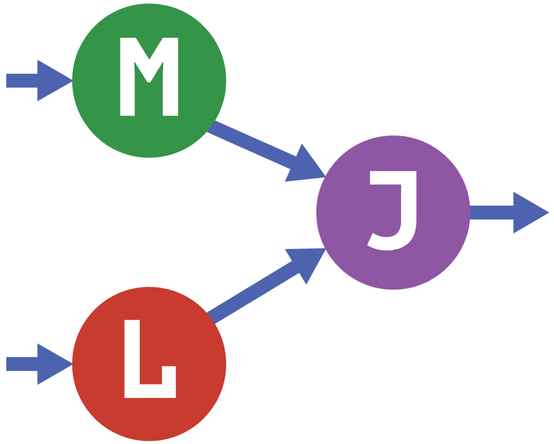
As an example, let's consider some popular ML packages: scikit-learn, tidymodels, h2o



User considerations:

- How "clean" is the API?
- Full data processing & ML pipeline, as well as visualizations
- Productionizing models
- Documentation & help (community)

Machine Learning in Julia: MLJ



MLJ is the most prominent meta-package for machine learning in Julia.

User considerations:

- How "clean" is the API?
- Full data processing & ML pipeline, as well as visualizations
- Productionizing models
- Documentation & help (community)

Modern ML: AutoML

Just as scikit-learn was an evolution in ML tooling, so is AutoML.

Auto-Sklearn

AutoML features:

- Automatically train & evaluate models
- Faster experiments with better reproducibility.
- Lowers barrier to entry
- Competitive performance

Modern ML: Explainability & Fairness

Another modern trend in machine learning is moving beyond simple model accuracy as a method to evaluate ML models.

🔬 Explainability:

- Black-box model explainers & visualizations

⚖️ Fairness:

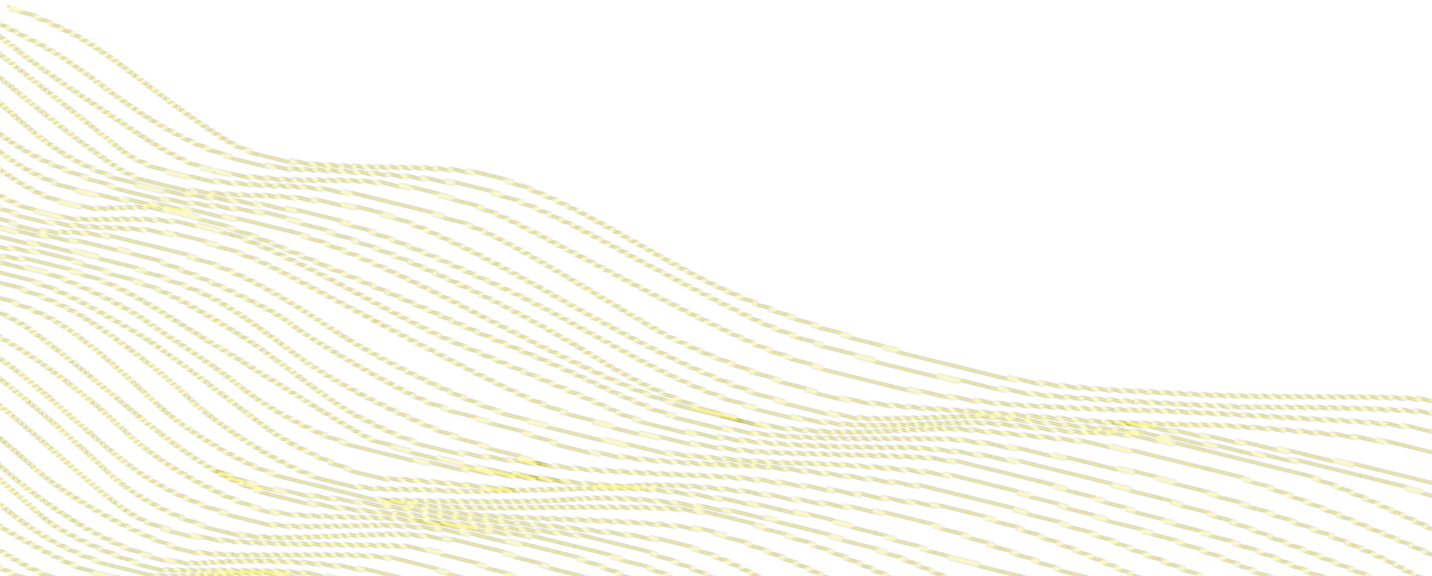
- measuring/mitigating the disparate impact on subsets of people (“protected” attributes such as age, gender, race, etc.)

Julia: ML To Do List

In order to compete with R and Python in data science and machine learning, Julia must keep progressing and meet the other machine learning platforms where they're at.

- AutoML
- Explainability
- Visualizations
- Fairness
- Production deployment
- MLOps

Community!



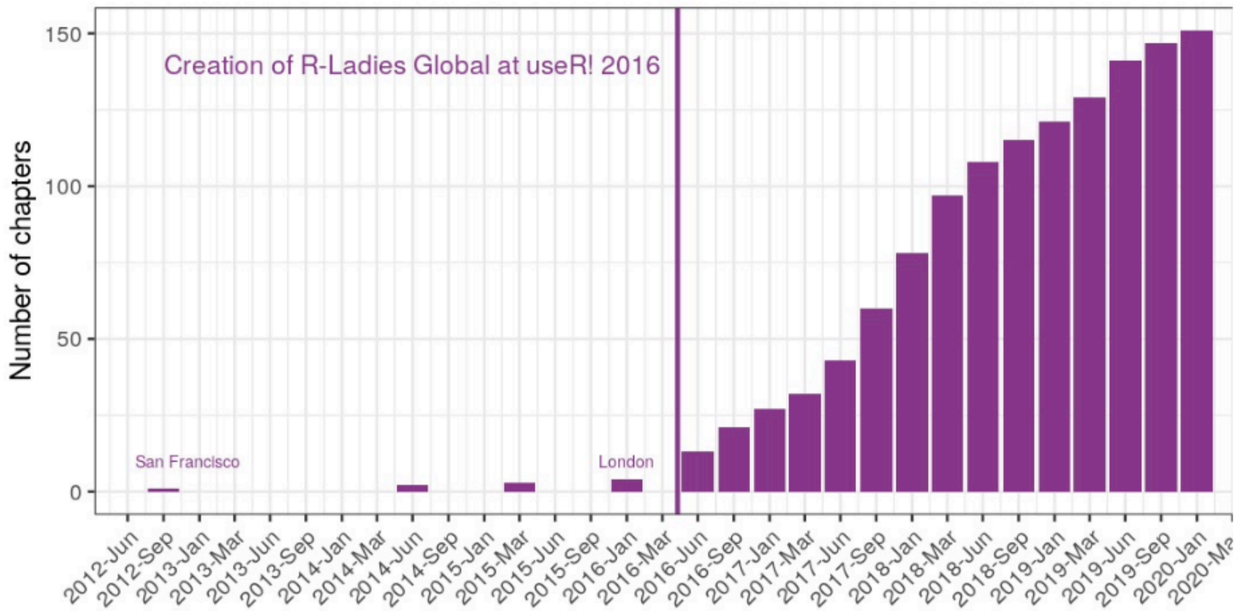
Community

In my opinion, community is the single most important aspect of the success of a language or software package.

Community relations:

- Package ecosystem
- OSS contributions
- Documentation & help
- Diversity & inclusion
- Day-to-day enjoyment
- Marketing/social

R-Ladies Global



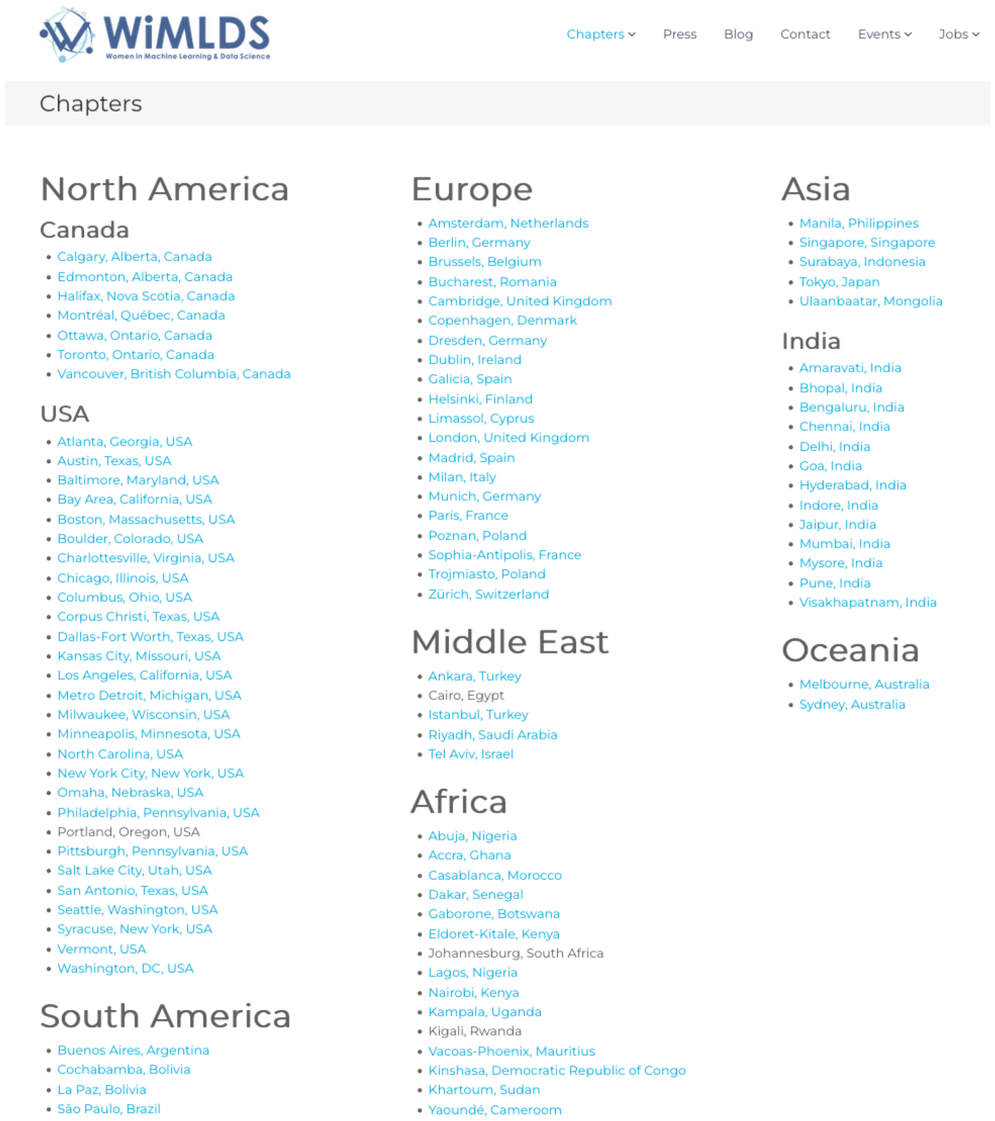
R-Ladies Global was formed at useR! 2016 and has become a driving force in the greater R community

- 88k members
- 218 groups
- Bigger than RUGs

https://rladies.org

@RLadiesGlobal on Twitter

WiMLDS



WiMLDS

Women in Machine Learning & Data Science

We have 100 Meetup chapters worldwide!

https://wimlds.org

@wimlds on Twitter

Julia Gender Inclusive

Julia Gender Inclusive was announced in June 2021!

Current activities:

- JuliaCon talk & BoF
- Online Meetups

My recommendations:

- IRL Meetups & local chapters when safe
- Record all meetup content & share on YouTube
- Private chat space
- Joint meetups with other diversity meetups

https://discourse.julialang.org/t/announcing-julia-gender-inclusive/63702

Community: To Do List

Developer inclusion:

- Run workshops to onboard underrepresented folks in contributing to Julia projects & core
- Add Julia Inclusive to Community page on julia.org
- Make Community a “top-level” priority of JuliaLang

Thank you!

@ledell on
Github, Twitter