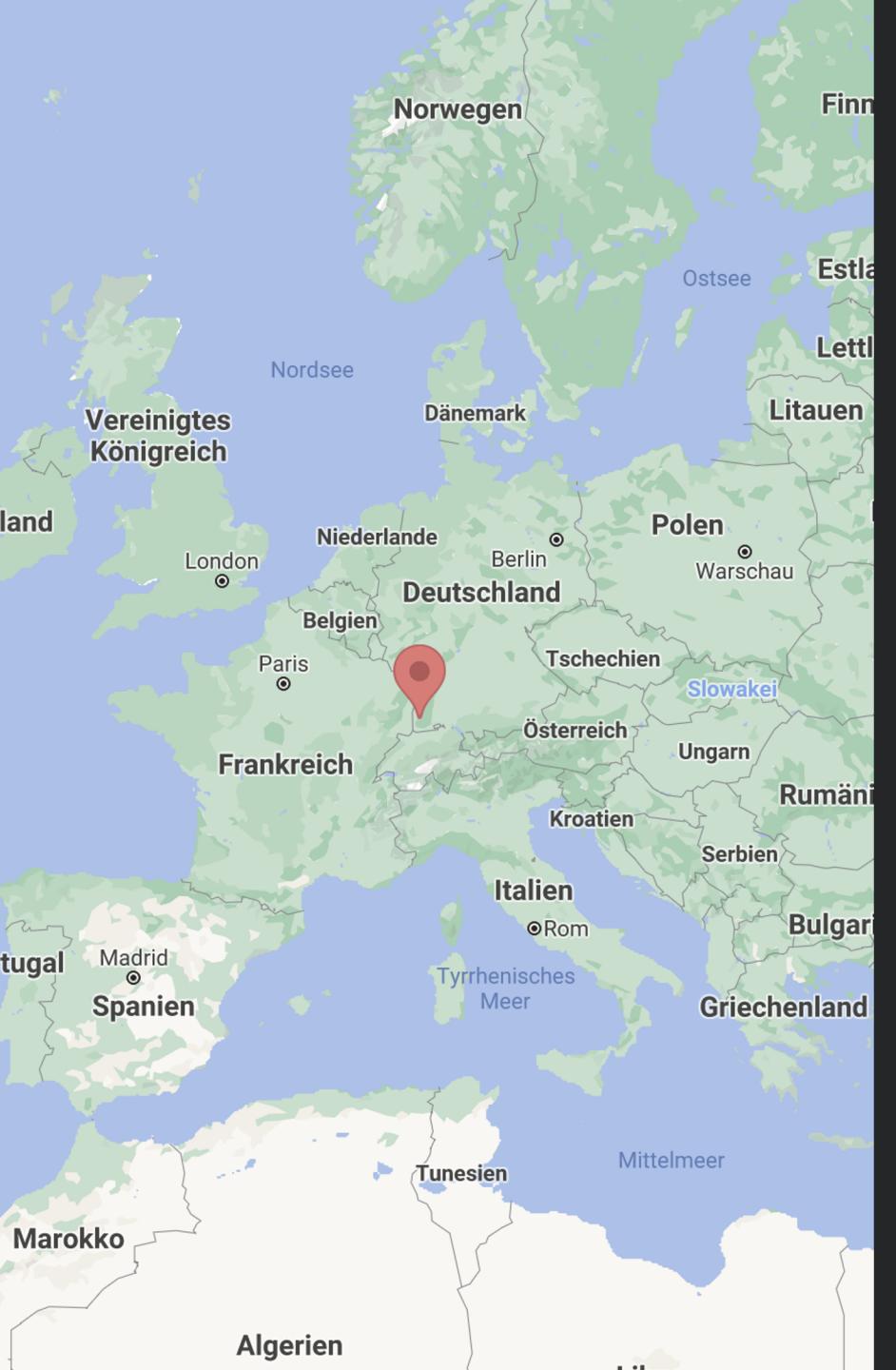


JULIA FOR R-LOVERS

Ideas for a dual-language workflow



ABOUT ME

Kyla McConnell

- PhD student in linguistics at the University of Freiburg
- R-stats enthusiast, Co-organizer of @RLadiesFreiburg
- Recently discovered the joys of Julia
 - *My use case: academic research, linear mixed effects models*

TODAY'S DEMO

What is Julia and why should you consider it?

Julia Basics, using R & Julia, and Julia for Data Science

Example: Fitting an LMM with MixedModels.jl

WHAT I WON'T COVER

01

Installing Julia, using
iJulia for Jupyter
Notebooks

- <https://www.youtube.com/watch?v=oyx8M1yoboY>

02

Programming concepts
in Julia -> **Julia Academy**

- <https://juliaacademy.com/p/julia-programming-for-nervous-beginners>
- <https://juliaacademy.com/p/intro-to-julia>
- <https://juliaacademy.com/p/julia-for-data-science>

03

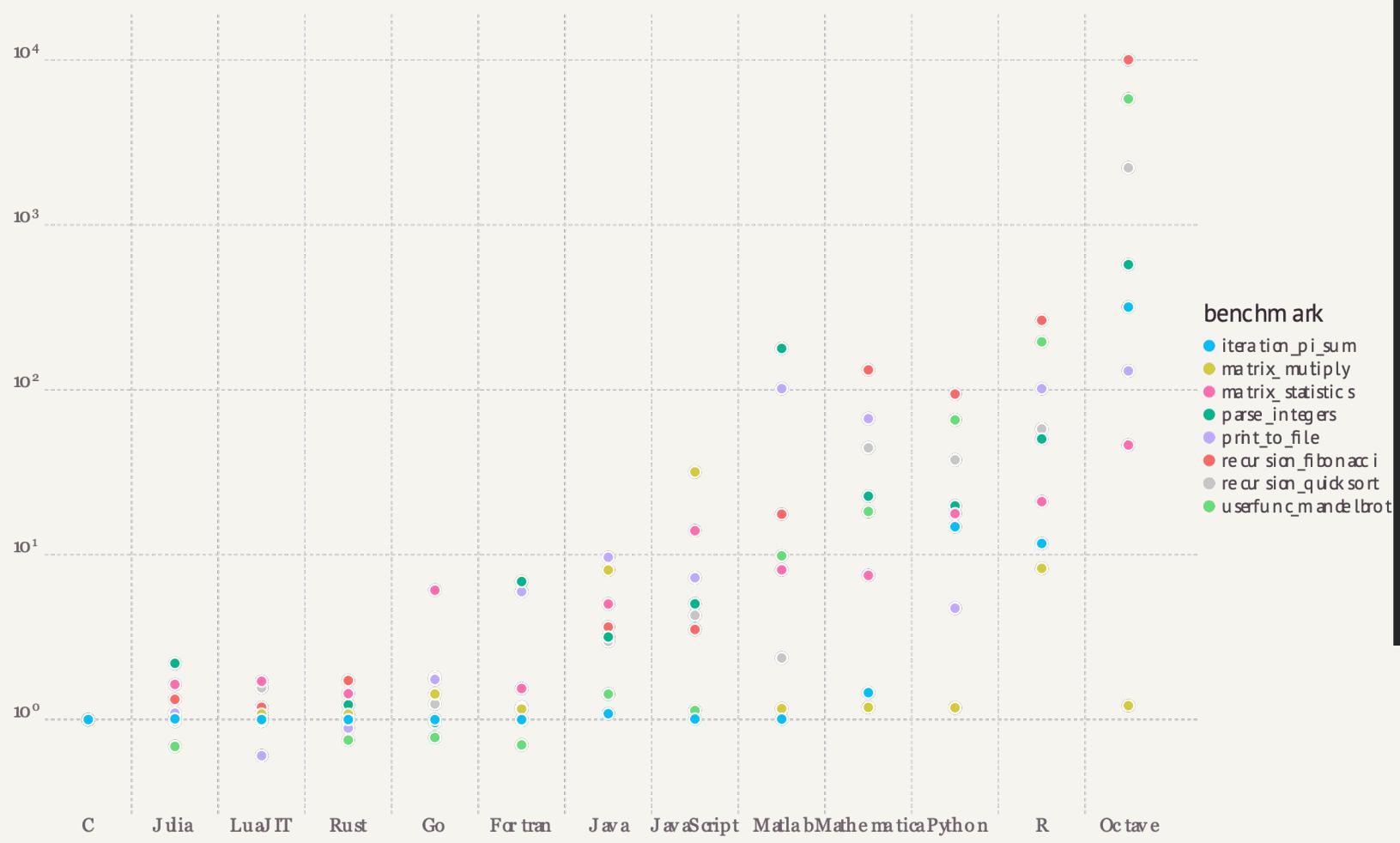
Statistical concepts or
packages

- <https://juliastats.org/MixedModels.jl/dev/>

WHAT IS JULIA?

- General-purpose programming language but popular in data science communities
 - *Built with data science and machine learning in mind*
- Free & open source (like R and Python)
- Developed to be readable like Python but compiled quickly like C
- Young: < 10 years old (R is closer to 30 years old)





“These micro-benchmarks, while not comprehensive, do test compiler performance on a range of common code patterns, such as function calls, string parsing, sorting, numerical loops, random number generation, recursion, and array operations.”

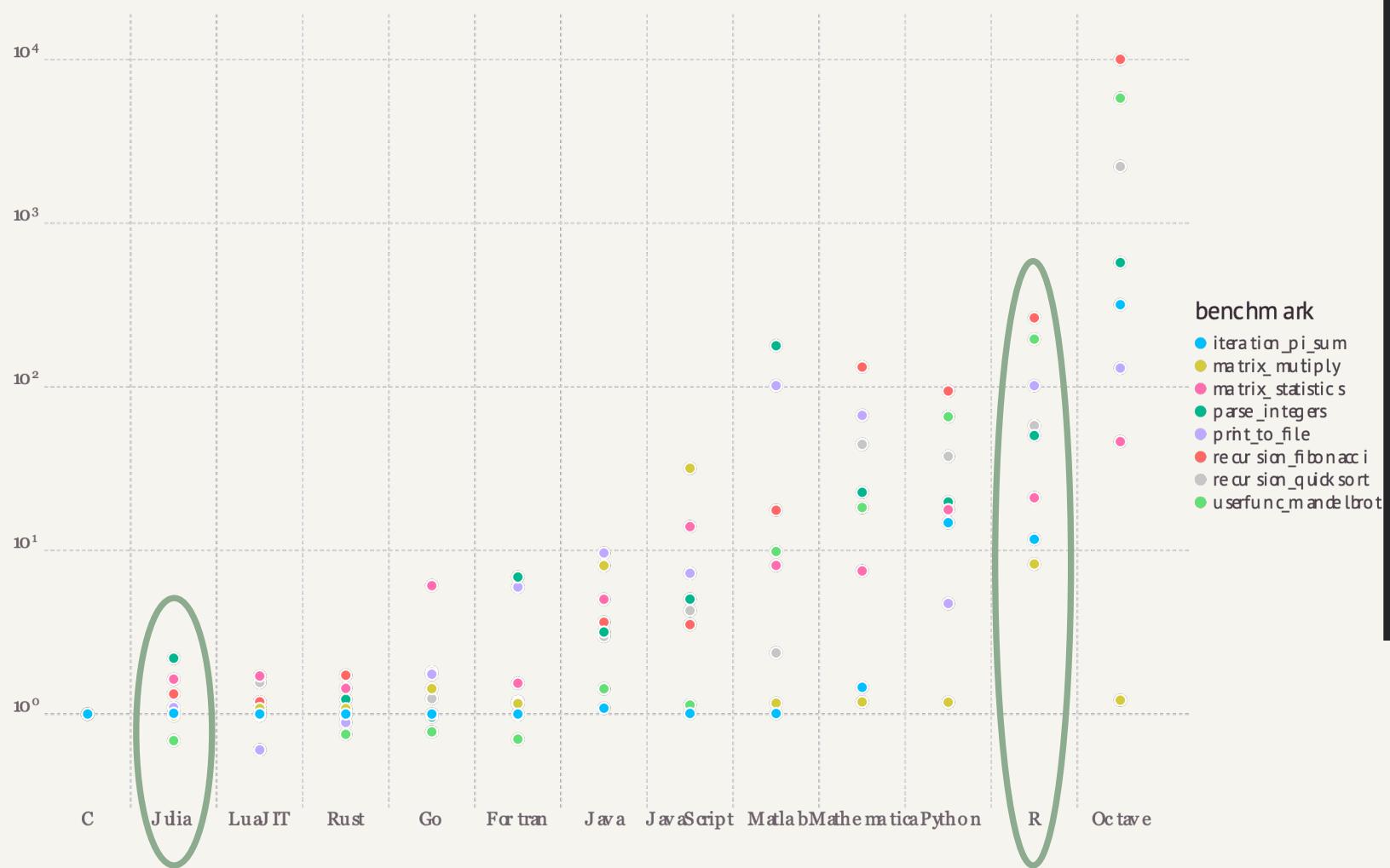
single core, Julia 1.0.0

<https://julialang.org/benchmarks/>

WHY JULIA?

- Gains in:
 - *Speed (but just-in-time compiler, i.e. package loading)¹*
 - *Model complexity*
- Without sacrificing:
 - *Readability*
 - *Reproducibility*
 - *R packages that you know well*
 - *Tutorials and resources from the R community*
 - *Ability to share code with colleagues who don't use Julia*

1 : <https://www.youtube.com/watch?v=FJeSa0Fr5VY>



“These micro-benchmarks, while not comprehensive, do test compiler performance on a range of common code patterns, such as function calls, string parsing, sorting, numerical loops, random number generation, recursion, and array operations.”

single core, Julia 1.0.0

<https://julialang.org/benchmarks/>

WHY JULIA?

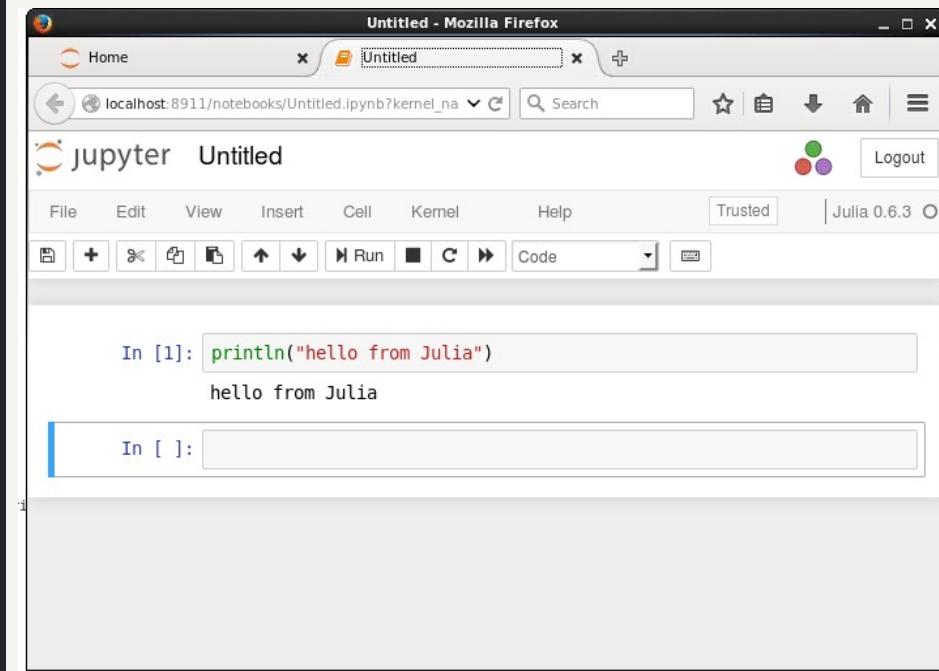
- Gains in:
 - *Speed (but just-in-time compiler, i.e. package loading)¹*
 - *Model complexity*
- Without sacrificing:
 - *Readability*
 - *Reproducibility*
 - *R packages that you know well*
 - *Tutorials and resources from the R community*
 - *Ability to share code with colleagues who don't use Julia*

1 : <https://www.youtube.com/watch?v=FJeSa0Fr5VY>

WORKING WITH R IN JULIA

■ Options:

- *REPL*
- *Julia script .jl*
- *Julia Markdown .jmd*
- *IDE like Visual Studio Code*
- *Jupyter Notebooks .ipynb*
- *Pluto Notebooks*



A screenshot of a Jupyter Notebook interface running in Mozilla Firefox. The title bar says "Untitled - Mozilla Firefox". The main window shows a "jupyter Untitled" tab. The toolbar includes File, Edit, View, Insert, Cell, Kernel, Help, Trusted, and Julia 0.6.3. A code cell is active, showing the command `In [1]: println("hello from Julia")`. The output below it shows the text "hello from Julia". There is another empty code cell below it labeled "In []:".



A screenshot of a Julia REPL interface. The top right corner displays the version information: "Documentation: https://docs.julialang.org", "Type "?" for help, "]?" for Pkg help.", "Version 1.5.1 (2020-08-25)", and "Official https://julialang.org/ release". Below this, the Julia prompt `julia>` is followed by the command `using RCall`. An R prompt `R>` is shown at the bottom, indicating a mixed R-Julia session.