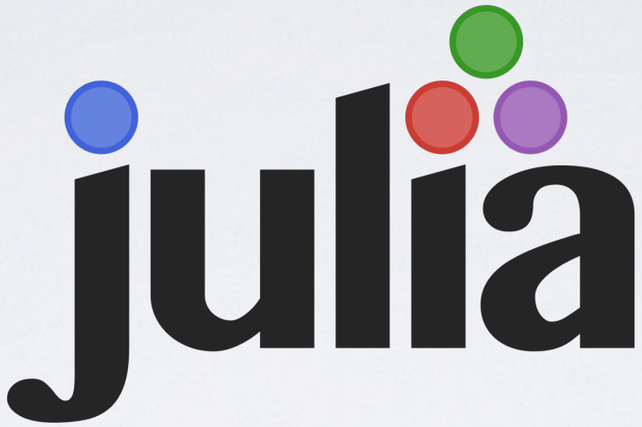
ESCAPE
European Science Cluster of Astronomy &
Particle physics ESFRI research Infrastructures

ERLANGEN CENTRE
FOR ASTROPARTICLE
PHYSICS



Tamás Gál (Tom)

tamas.gal@fau.de

Erlangen Centre for Astroparticle Physics (ECAP)

tamasgal.com

https://github.com/tamasgal

@tamasgal

WHAT IS JULIA?

JULIA

https://julialang.org/downloads/

- A **high-level**, **high-performance dynamic** programming language **specifically designed for science and numerical computing**
- **Multi-paradigm language** (combines features of procedural, functional, meta and object-oriented programming), while **multiple dispatch** is the primary one
- A **huge part** of the language is **written in Julia** itself!
- Optional typing (**type inference**)
- just-in-time (JIT or also called as JAOT: just-ahead-of-time) compilation (LLVM)
- The **performance** approaches that of **statically-compiled languages** like C
- **Lisp-like macros** and other meta-programming features
- **Call C-functions** directly (no wrappers or special API needed)
- **Interfaces with other languages** through packages like PyCall, RCall, etc.

WHY JULIA?

- You often write your **own algorithms** or want to **modify existing** ones (which are written in Julia as well)
- You want to write **prototype** code, which is usually **as fast as optimised** code
- You want to **learn something different** (always recommended!)
- You want zero overhead calls to e.g. **C** and **Python**
- You want to use **all your CPUs and GPUs** and even **multiple machines in distributed computing**
- You love working in **Ju**pyter (the **Ju** stands for **Ju**lia)
- The community is **full of scientists**! ;)

PYTHON, ITS POPULARITY AND FUNDAMENTAL ISSUES

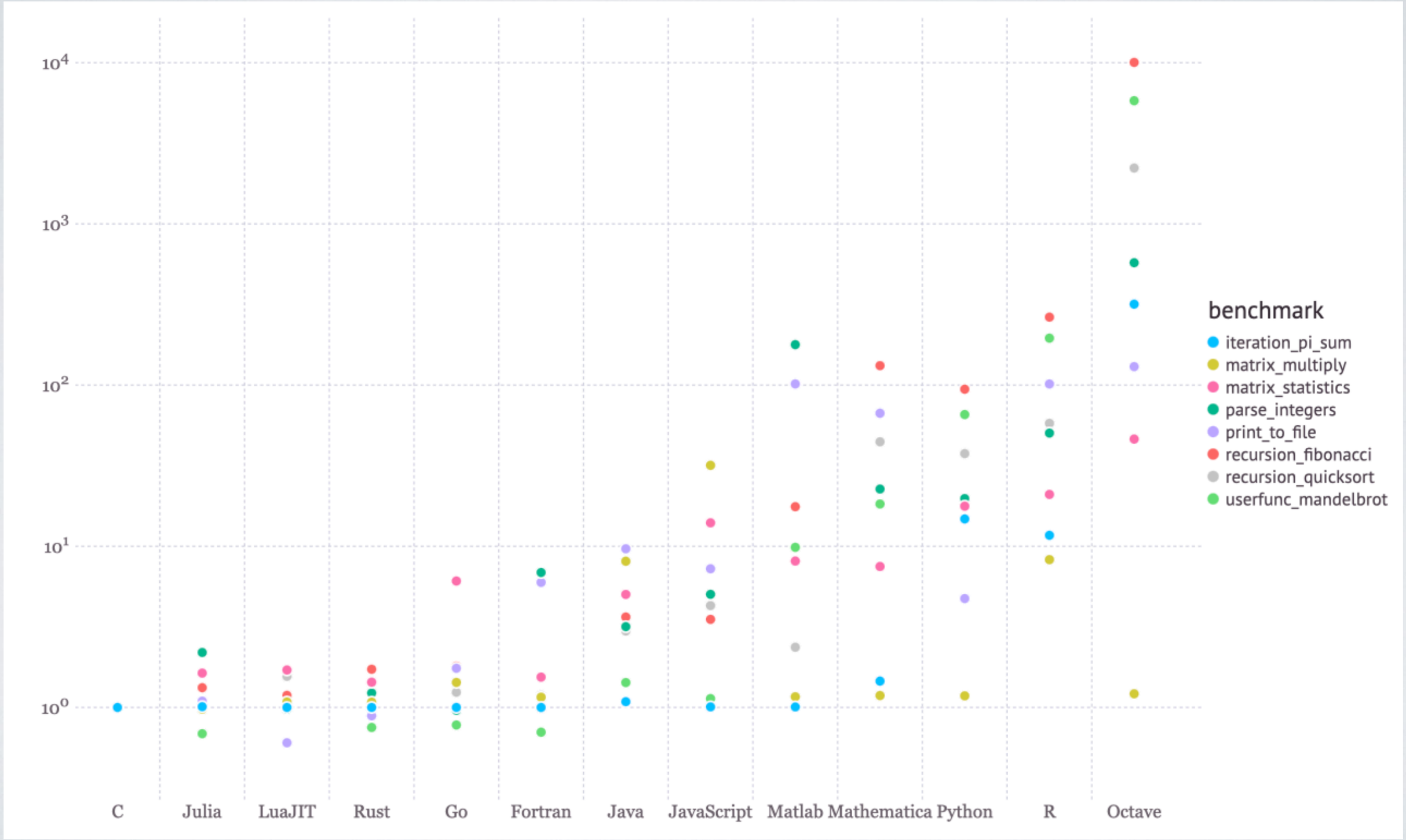
Partly opinionated view, but shared by many scientists out there...

- **Python is already installed** (too often it's still version 2.7 which is desperately outdated, but still ...) on most systems by "default"
- **It's easy to use** (every student can manage to **quickly read a dataset** and **create a plot** within a few minutes, from scratch)
- **Many packages** and **libraries** out there to do whatever comes to your mind
- **NumPy** made it easy to crunch numbers and Machine Learning is Python-dominated
- **Python** is **not** a language designed for **scientific** computing
- Users are often only facing a **high-level API** with **limited control over the internals**
- ... you need to learn a **lot of different technologies** to achieve **high performance**
- Developers spend **huge amounts of time and effort** to create and interface high performance libraries to **Python** (numpy, pandas, Tensorflow, Keras, PyTorch, h5py, pytables, AwkwardArray) and even different implementations of the Python interpreter/ compiler itself (PyPy, IronPython, Cython, Jython, Numba ...)

JULIA PACKAGES?

- **Plenty** of **scientific** (and other) packages available in the **Julia package registry**
- A great site to explore those: **https://juliapackages.com**
- **JumMP.jl** modeling language for mathematical optimisation (linear, mixed-integer, conic, nonlinear ...)
- **Flux.jl** for machine learning (entirely written in Julia!) 3k GitHub stars, last commit a few hours ago ...
- **DifferentialEquations.jl** for high-performance solvers of differential equations and scientific machine learning components
- **IJulia.jl** to install the Julia kernel for Jupyter
- **Plots.jl** the plotting library with many backends (including Matplotlib ;)
- **DataFrames.jl** In-memory tabular data in Julia, inspired by R-DataFrames (which also inspired Pandas in the Python world)
- **Zygote.jl** source-to-source automatic differentiation (AD) in Julia, the next generation AD system for Flux.jl (see above)
- **Bat.jl** a Bayesian analysis toolkit

SOME BENCHMARKS



Taken from https://julialang.org/benchmarks/

LIVE DEMO

https://julialang.org/downloads/

THANK YOU!