
# Julia in HEP

Graeme A Stewart, Benedikt Hegner, Pere Mató - CERN EP-SFT

Thanks also to Tamas Gal


GDB, 10 January 2024

# Julia - what's that then?


- The Julia language was launched in 2012 (v1.0 in 2018)
  - New, but not immature!
- Modern imperative language, multi-paradigm with reflection and object orientation
- Robust built-in tooling (learning from earlier languages)
  - Outstanding integrated package manager and build system
  - Module system with excellent code reuse
  - Modern tooling, with built in debuggers and profilers
  - Interactive - REPL and full notebook support (it's the "Ju" in Jupyter)
- Julia has been built from the ground up to be **very fast**
  - JIT compilation via LLVM to native machine code
  - Performance is comparable to C and C++ (as a baseline, see microbenchmarks)

# Solving the *Expression Problem*

- For code reuse we want to…
  - Define new types to which existing operations apply
    - Easy in OO, hard in functional
  - Define new operations that apply to existing types
    - Hard in OO, easy in functional
- Julia uses the paradigm of multiple-dispatch
  - Generic programming allows different parts of code to mix, JIT keeps everything fast
  - Add new methods to existing generic functions for new types ✅
  - Add new methods to generic functions for existing types ✅

*These packages know nothing about each other!*

**DifferentialEquations + Measurements**

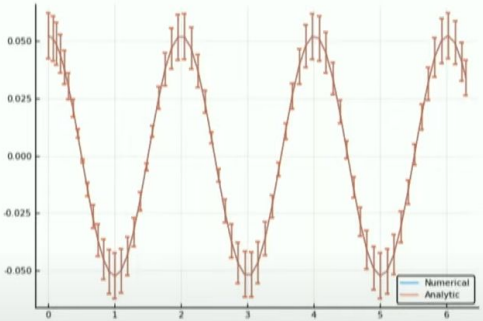
```
g = 9.79 ± 0.02 # Gravitational constants
L = 1.00 ± 0.01 # Length of the pendulum

# Initial speed & angle, time span
u₀ = [0 ± 0, π/60 ± 0.01]
tspan = (0.0, 6.3)

# Define the problem
function pendulum(du, u, p, t)
    θ  = u[1]
    dθ = u[2]
    du[1] = dθ
    du[2] = -(g/L)*θ
end

# Pass to solvers
prob = ODEProblem(pendulum, u₀, tspan)
sol = solve(prob, Tsit5(), reltol = 1e-6)

# Analytic solution
u = u₀[2] .* cos.(sqrt(g/L) .* sol.t)
```

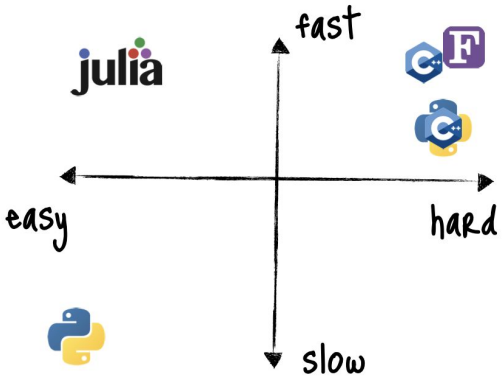



Rackauckas et al. DifferentialEquations.jl – A Performant and Feature-Rich Ecosystem for Solving Differential Equations in Julia. 2017. (Journal of Open Research Software)

Giordano. Uncertainty propagation with functionally correlated quantities (arXiv: 1610.08716)

JuliaCon 2019 | The Unreasonable Effectiveness of Multiple Dispatch | Stefan Karpinski
https://www.youtube.com/live/kc9HwsxE1OY

# But is Julia interesting for HEP?

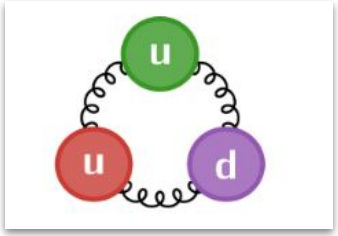



- There exist many languages in the world
  - Each has different strengths and weaknesses
- We think the answer is yes!
  - Julia is specifically designed for numerical programming for science and engineering*
    - So we are the target audience and the support for our use case is strong
  - Julia is much easier to program in than C++
    - Experience shows that students with Python experience can be productive in Julia very quickly
  - Code written in Julia is fast, often close to peak performance
    - The first prototype can evolve naturally into the production code
  - This overcomes the two language problem that we have today
    - We use Python because it's human productive, we use C++ because it's fast
    - We suffer from friction at the interface, plus a general diminishing of skills and interest in C++
  - Wrappers allow integration with existing code in C++ and Python - vital for our existing codes
  - Interfaces needed to ease the use of Julia in HEP are maturing

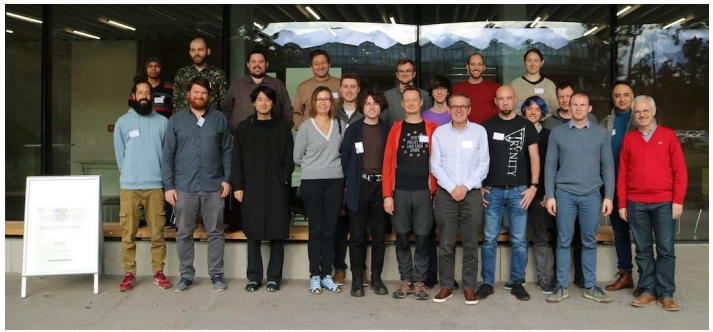
*Julia used a lot in ASML, Boeing, Pfizer among others

# JuliaHEP

- New HSF working group created after CHEP 2023 to channel community efforts to evaluate and introduce Julia language in HEP
  - A paper on the Potential of the Julia Programming Language for High Energy Physics Computing was in the pipeline at this time and is now published [Comput Softw Big Sci 7, 10 (2023)]

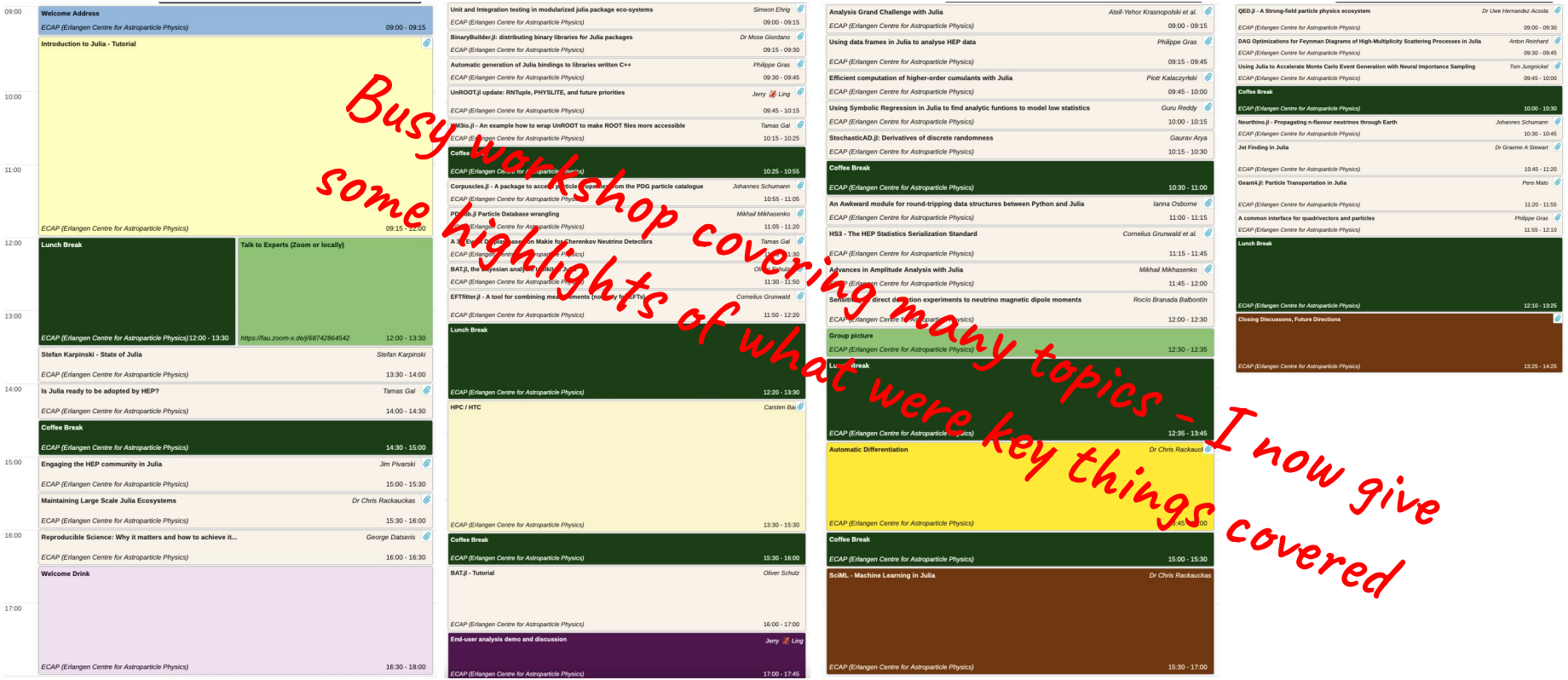



- First JuliaHEP workshop organised at ECAP in Erlangen, Germany
  - 4 full days (6 to 9 of November)
  - 20 people in person + ~30 people remote
  - An agenda organized with tutorials, keynote presentations, long and short presentations
  - Contributions from key Julia developers (including one of the language founders, Stefan Karpinski)
  - Plenty of time for discussions



Erlangen Centre for Astroparticle Physics (ECAP)

# Agenda

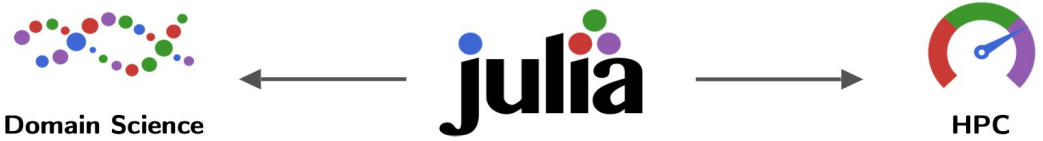



Busy workshop covering many topics - I now give some highlights of what were key things covered

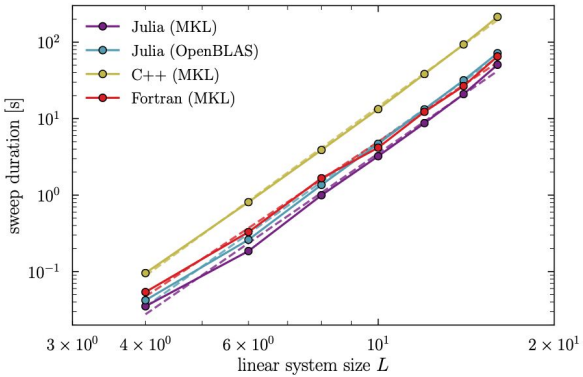
# JuliaHEP WS: Tutorials

- **Introduction to Julia** - Sam Skipsey and Graeme Stewart
  - Intended for beginners to get started (language basics, multidimensional arrays, functions, multiple dispatch,  plotting, dataframes, etc.)
  - New training material developed for the occasion
    - Presented as Jupyter Book (collection of notebooks)
    - Available also in the HSF Software Training Center
- **Julia for High-Performance Computing (HPC)** - Carsten Bauer
  - After a 3 short introductory talks on Julia on  HPC (Carsten Bauer, Mose Giordano, Ludovic Räss) we had a hands-on tutorial using the Noctua 2 cluster (143.872 cores)
  - https://github.com/carstenbauer/juliahep-hpctutorial
  - Provided login credentials to all interested participants
- **BAT.j: Bayesian Analysis Toolkit** - Oliver Schulz
  - Hands on tutorial

# HPC




- Julia can be a great option for HPC!
  - **Serial and parallel performance on-par with Fortran/C/C++**
  - **portability and high-productivity** (same julia packages on Laptop and HPC clusters)
  - **New opportunities** (e.g. interactive HPC)
- Challenges
  - Julia depot (downloaded packages and artifacts) can get under (a lot) of pressure!
  - Memory footprint, O(1 GB) per process
- Examples HPC projects
  - CliMA @ Caltech - Climate Modeling Alliance
  - CESMIX @ MIT - Exascale simulation of materials in extreme environments
  - Trixi @ RWTH Aachen / HLRS - Computational fluid dynamics
  - GPU4GEO @ ETH / CSCS - Computational earth science



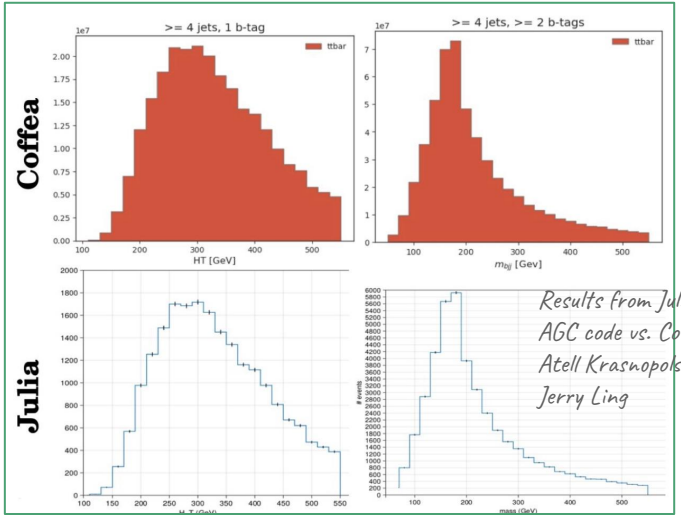

**QMC**
(MonteCarlo.jl)

# Tooling

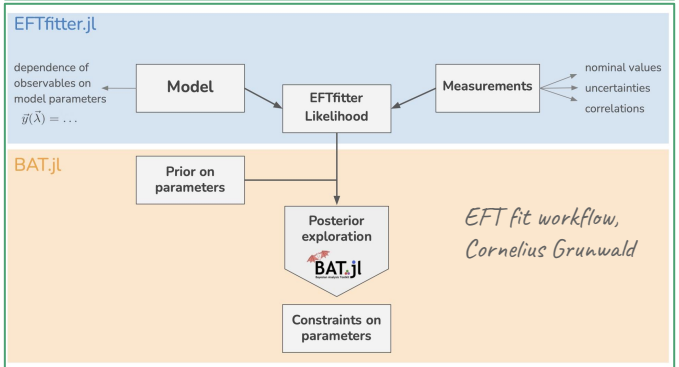
- BinaryBuilder.jl
  - To produce and pack binaries as Julia packages (**_jll** packages) for large combinations of architecture/operating system/compiler and resolving all dependencies
- UnROOT.jl
  - Julia package to read ROOT **TTrees** and **RNTuples**
  - Implements **Tables.jl** interface with the **LazyTree** to read the requested column when needed. It is fast and multi-threading friendly
- WrapIt
  - Automatically generate code to wrap C++ packages (**CxxWrap.jl**)
- Unit and Integration testing
  - Native support for unit tests and easy deployment of integration tests
- Scientific project reproducibility
  - Practical guidelines to achieve reproducibility
  - Each project comes with two essential files for **Project.toml** (direct dependencies and version requirements) and **Manifest.toml** (the exact version of all dependencies)

# Data Analysis

- Analysis Grand Challenge
  - Columnar data extraction, filtering, new columns, systematic variations into histograms. Statistical model construction and visualisation.
  - Equivalent results as Coffea and PyHEP tools. Loop with < 100 lines. Good scaling on 25 nodes.
- Julia DataFrames Analysis
  - Demonstrated that analysis can also be done with DataFrames.jl (equivalent to Pandas in Python)
- Corpuscles.jl & PDGdb.jl
  - Modules to provide easy access to Particle Data (PDG)
- BAT.jl and EFTFitter.jl
  - Powerful tools for Bayesian analysis and to constrain free model parameters by combining multiple measurements

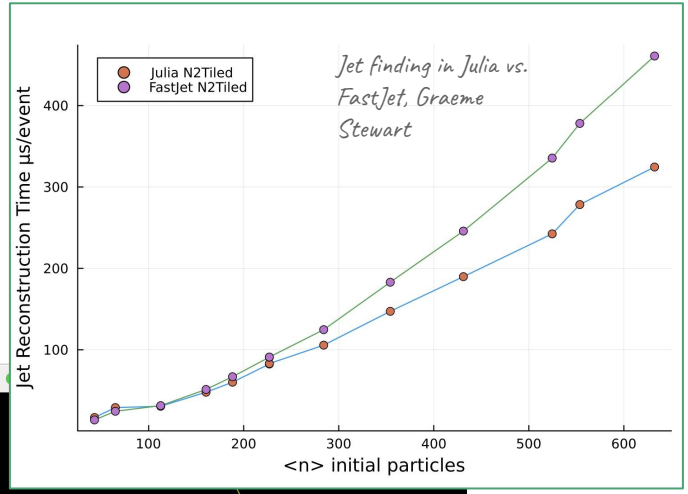


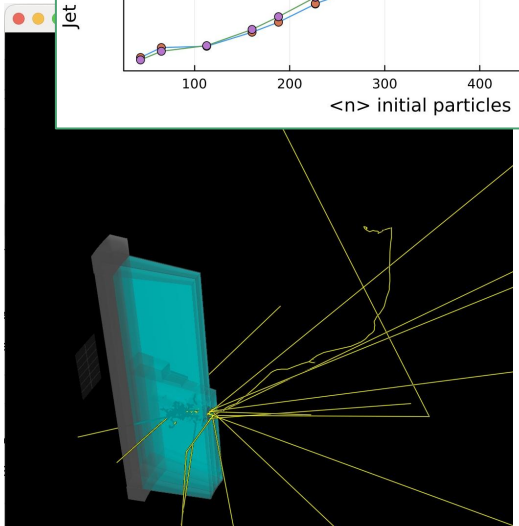

| | Julia DF | Julia Loop | Coffea |
|---|---|---|---|
| Execution ($t/t_{\text{fastest}}$) | 33 s (1.2) | 27 s (1) | 158 s (5.9) |
| JIT compilation | +4.9 s | +2.2 s | – |
| Mem. allocation | 40 GiB | 19 GiB | – |



# MC Generator, Simulation and Reconstruction

- QED.jl - Strong-field particle physics ecosystem
  - Fields, processes, phase space, event generation, etc.
  - Optimizations for large number of Feymann diagrams
  - Neural importance sampling (better than classical VEGAS)
- Jet Finding
  - Test-case for algorithm development and testing ergonomics
  - Julia reaches C++ speed
- Geant4.jl
  - Wrapping C++ Geant4 with a new simpler and more ergonomic API
  - MT support and very good performance
- Neurthino.jl
  - Neutrino oscillation probability calculator

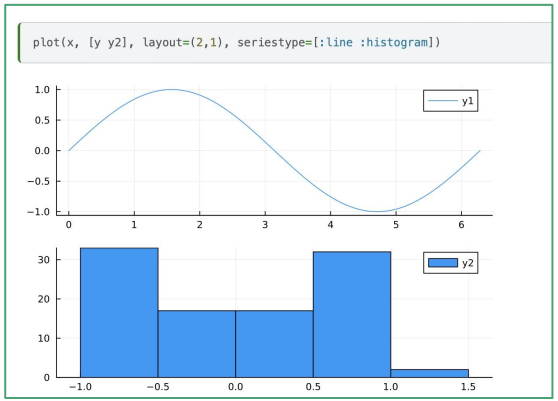


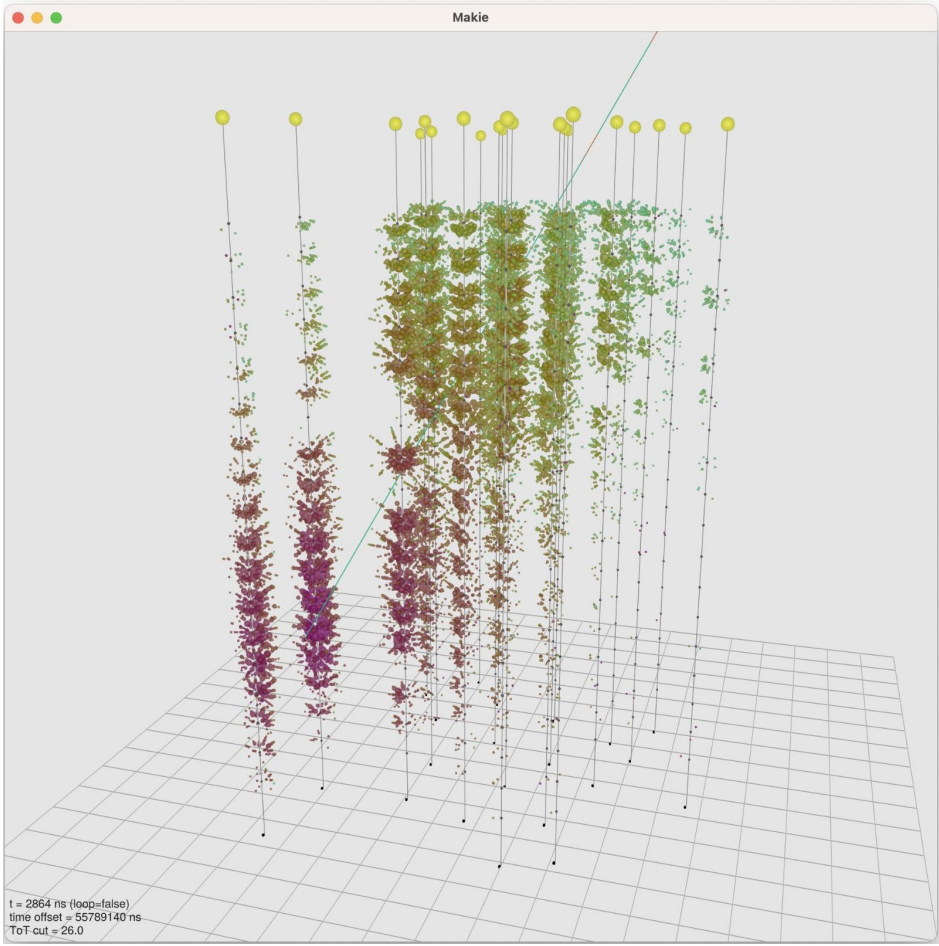
Geant4 generated event using Geant4.jl wrappers, Pere Mato

# Visualization

- Two major packages for data visualization: **Plots.jl** and **Makie.jl**
  - Plots.jl is probably easy to use out of the box, while Makie.jl is more powerful but requires some learning




- Makie Example: Event display for Neutrino Detectors (KM3NeT)

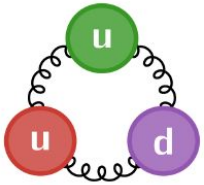



Display from RainbowAlga.jl, Tamas Gal, ECAP

# Automatic Differentiation and Scientific ML

- Chris Rackauckas (Julia Lab Co-PI) gave a masterclass on *Differentiable Simulation* (using the whiteboard and advanced calculus)
- Impossible to reproduce here :-)
- Main messages we got:
  - Add a NN only on the unknown parts of the ODE system. It gives much better predictivity
  - Julia has many packages for AD using different methods, from symbolic manipulation to operator overloading, in forward and reverse mode (https://juliadiff.org)

# What's Next?

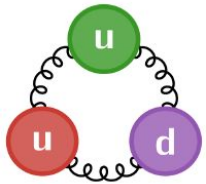



- We identified a number of important items to work on to improve integration into the HEP world
  - More wrappers to speak to existing code: HepMC3, Minuit, FastJet, Pythia8, etc.
  - The ability to *write* RNTuples (see an update from Jerry Ling on this important topic)
  - Develop better support for histograms in FHist.jl (or redo) and support statistical standards like HS3
  - Generic HEP support for Lorentz Vectors, etc, plus plotting recipes in Plots.jl and Makie.jl
- And we need to work on topics that make Julia deployable at the large scale
  - Precompilation of packages (we don't want to JIT on 1M nodes!)
  - Can leverage a lot of experience from the HPC community here

# What's Next?

- Julia for ML
  - Native packages for deep learning, relying on Julia's great strengths in autodifferentation and JIT speeds
- Julia on GPUs
  - CUDA, HIP, oneAPI supported
  - More interestingly the *kernel abstractions package* allows all of these specifics to be hidden
- Training
  - Keep developing material and presenting it!

HSF JuliaHEP WG now organising monthly meetings - lots to get involved in!




Next JuliaHEP workshop will be
30 September - 4 October at CERN