Julia for AGC

Atell-Yehor Krasnopolski

IRIS-HEP Fellow

Taras Shevchenko National University of Kyiv

Jerry Ling

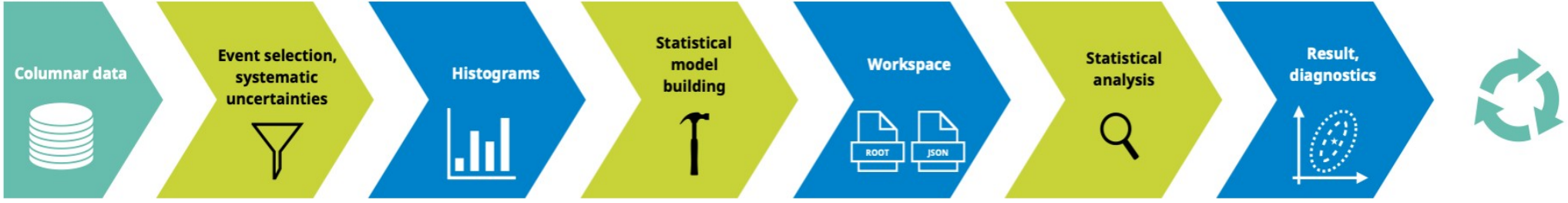
Harvard University

Alexander Held

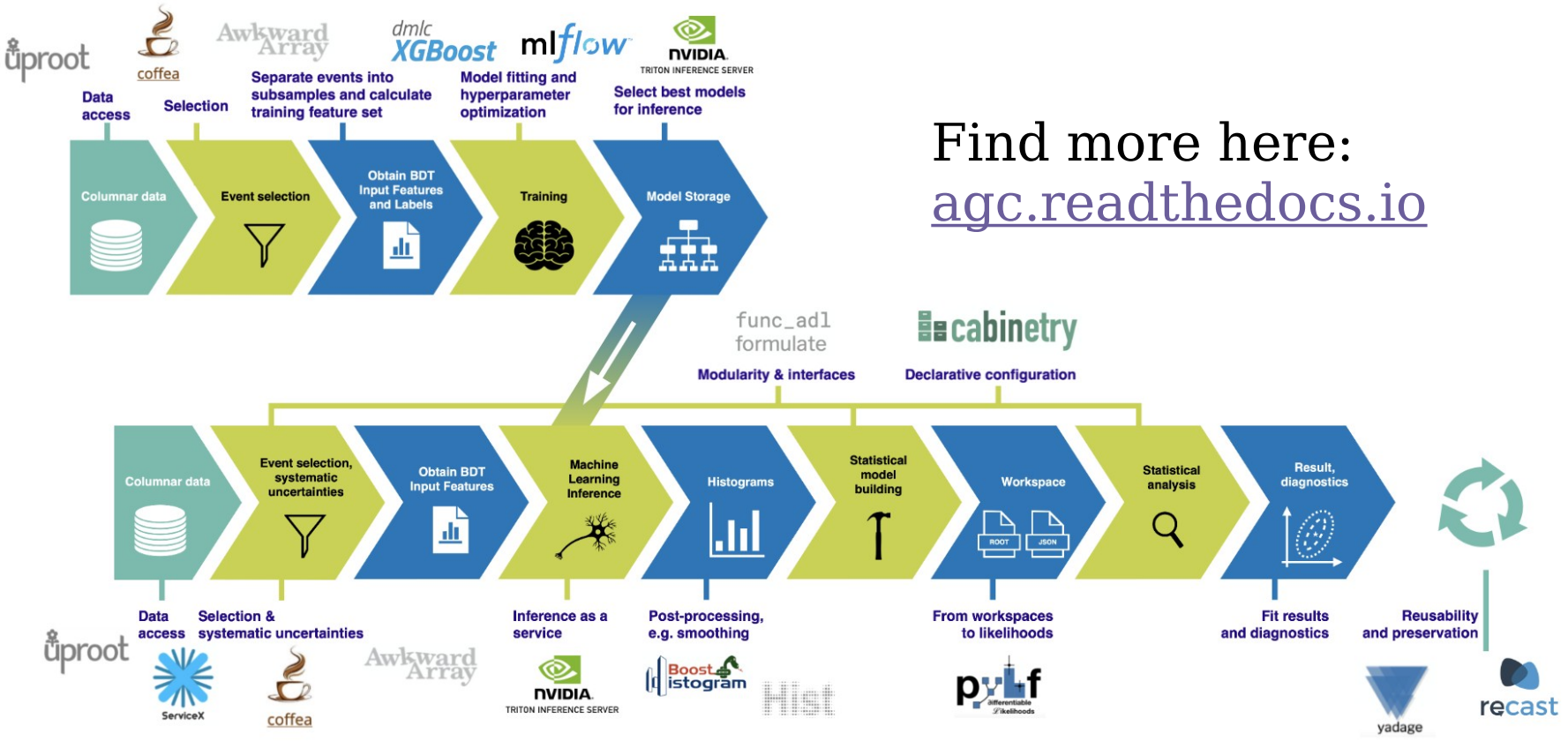
UWM

Analysis Grand Challenge

- columnar data extraction from large datasets
- processing of that data (event filtering, construction of observables, evaluation of systematic uncertainties) into histograms
- statistical model construction and statistical inference
- relevant visualisations for these steps



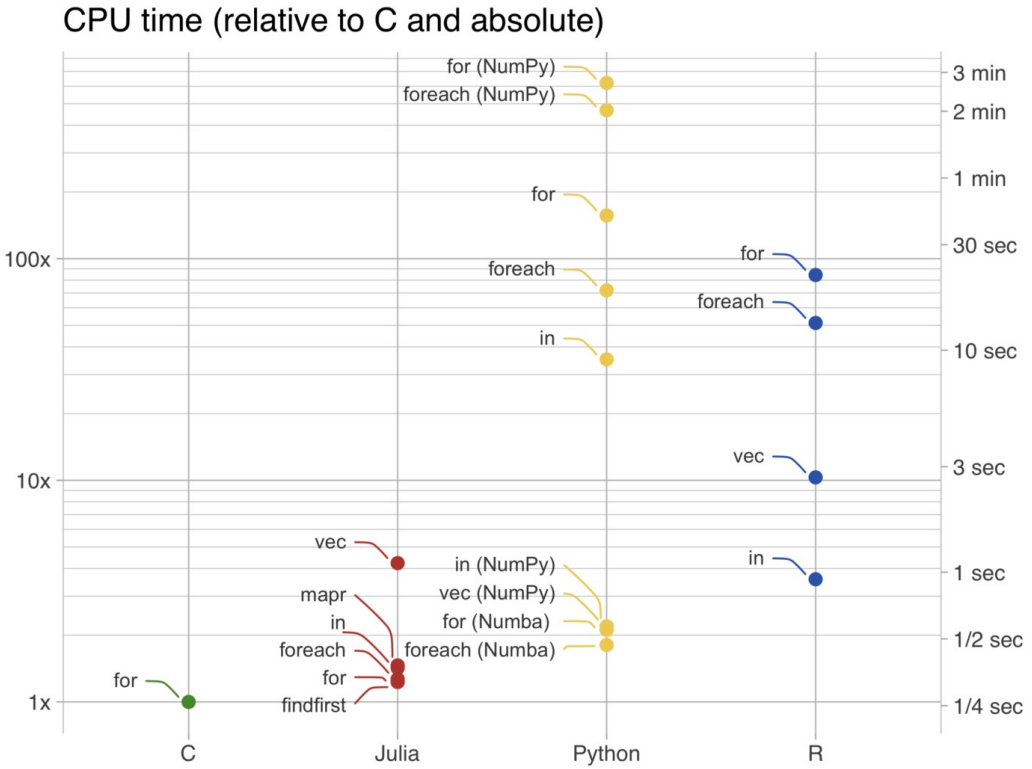
Analysis Grand Challenge



Find more here:
agc.readthedocs.io

Why julia

- Perfect for Physics and Mathematics
- Fast by design, not because of packages & JIT-compiled
- Can interact with C, FORTRAN & Python
- Proven to be efficient for HEP: github.com/JuliaHEP
- www.juliahep.org
- arxiv.org/abs/2306.03675



(towardsdatascience.com/r-vs-python-vs-julia-90456a2bcbab)

julia for this task

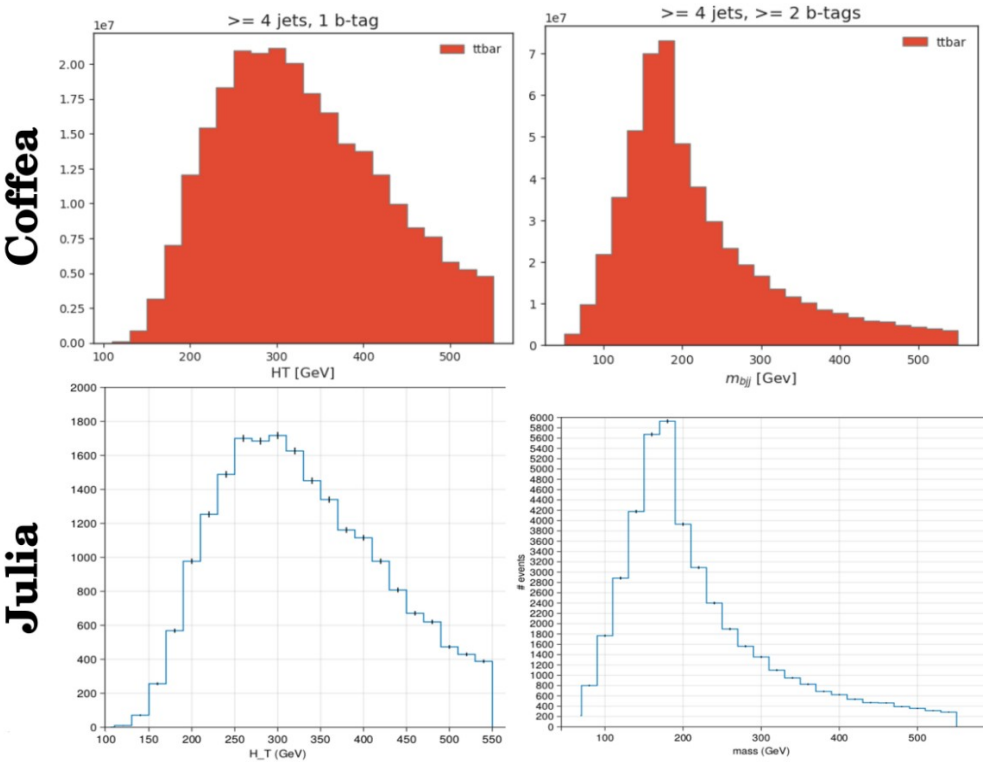
- Less than 100 lines of code for the main loop
- Plotting, distributed computing, and working with complex data structures could not be easier
- Syntax & general experience

- Some tools were a bit raw when we started

```
function get_histo(tree, wgt; file_variation::Symbol=:nominal, evts=nothing)
    is_nominal_file = (:nominal == file_variation)
    hists = generate_hists(file_variation)
    #threads_gtthreads for evt in tree
    for evt in tree
        # single lepton requirement
        (; Electron_pt, Muon_pt) = evt
        (count(>(25), Electron_pt) + count(>(25), Muon_pt) != 1) && continue

        # get pt
        (; Jet_pt) = evt
        is_nominal_file && (Jet_pt_nominal = Jet_pt)

        for hist_type in (is_nominal_file ? keys(SHAPE_VARS) : (:nominal,))
            # modify pt
            is_nominal_file && (Jet_pt = SHAPE_VARS[hist_type](Jet_pt_nominal))

            scale_info = (; Jet_pt, wgt)

            # at least 4 jets
            jet_pt_mask = Jet_pt .> 25
            if count(jet_pt_mask) >= 4
                jet_btag = @view evt.Jet_btagCSVV2[jet_pt_mask]

                btag_count = count(>(0.5), jet_btag)
                # MASS_HISTOGRAM
                if btag_count >= 2 # at least 2 btag
                    if evts !== nothing
                        if hist_type in keys(evts)
                            push!(evts[hist_type], evt.event)
                        end
                    end

                    (; Jet_eta, Jet_phi, Jet_mass) = evt

                    # construct jet lorentz vector
                    jet_p4 = @views LorentzVectorCyl.(Jet_pt[jet_pt_mask], Jet_eta[jet_pt_mask], Jet_phi[jet_pt_mask], Jet_mass[jet_pt_mask])

                    Njets = length(jet_btag)

                    # tri_jet combinatorics
                    max_pt = -Inf
                    local best_mass

                    # 1. all tri-jet combinations
                    for comb in Combinations(Njets, 3)
                        p4s = @view jet_p4[comb]
                        btags = @view jet_btag[comb]
                        # 2. keep those maximum(btags1,2,3) > 0.5
                        maximum(btags) <= 0.5 && continue
                        tri = sum(p4s)
                        _pt = pt(tri)
                        # 3. pick the tri-p4 with highest tri-pt
                        if _pt > max_pt
                            max_pt = _pt
                            best_mass = mass(tri)
                        end
                    end

                    # tri-p4 with highest tri-pt first
                    push!(hists[Symbol(:mbjj_4j2b_, (is_nominal_file ? hist_type : file_variation))], best_mass, wgt)
                    if is_nominal_file && (hist_type == :nominal)
                        @scale_var_loop :mbjj_4j2b best_mass
                    end
                # HT HISTOGRAM
                elseif btag_count == 1 # no more than 1 btag
                    HT = @views sum(Jet_pt[jet_pt_mask])
                    push!(hists[Symbol(:HT_4j1b_, (is_nominal_file ? hist_type : file_variation))], HT, wgt)
                    if is_nominal_file && (hist_type == :nominal)
                        @scale_var_loop :HT_4j1b HT
                    end
                end
            end
        end
    end

    hists
end
```

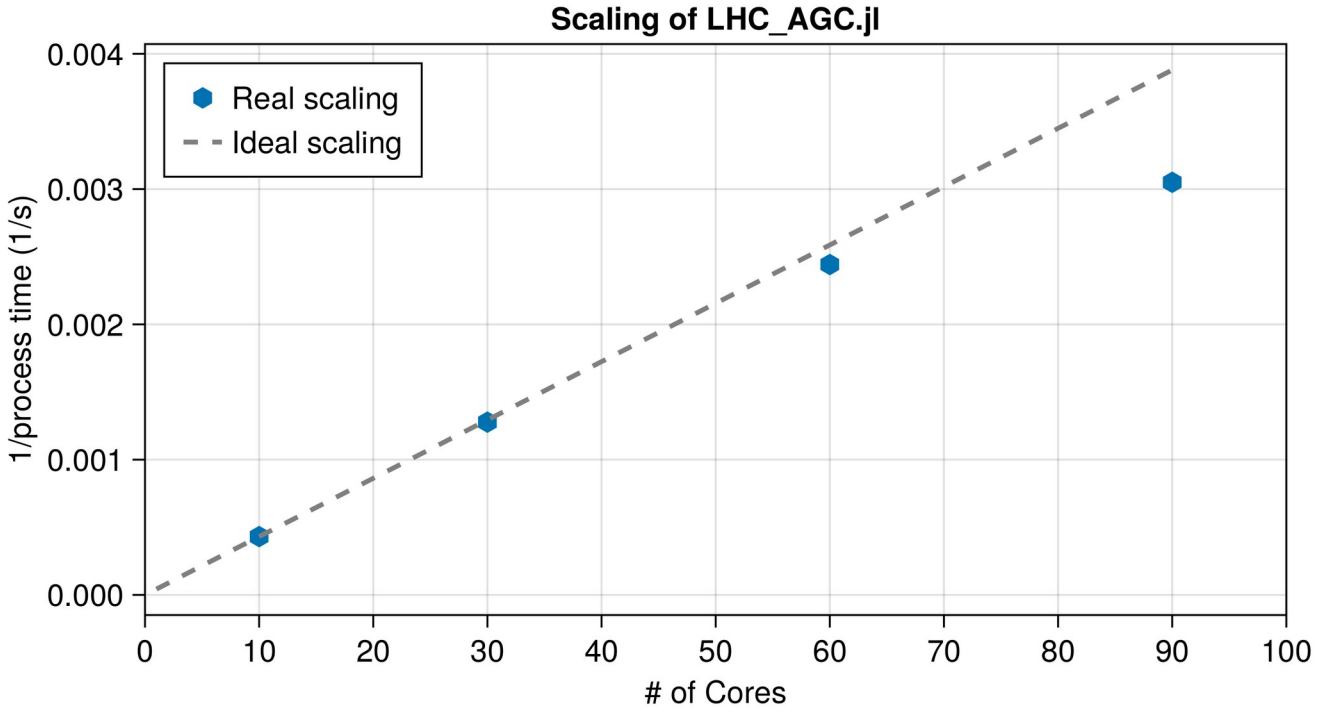
Results

- The whole pipeline (except ML-related parts)
- Generating correct histograms with native Julia up to bin migrations



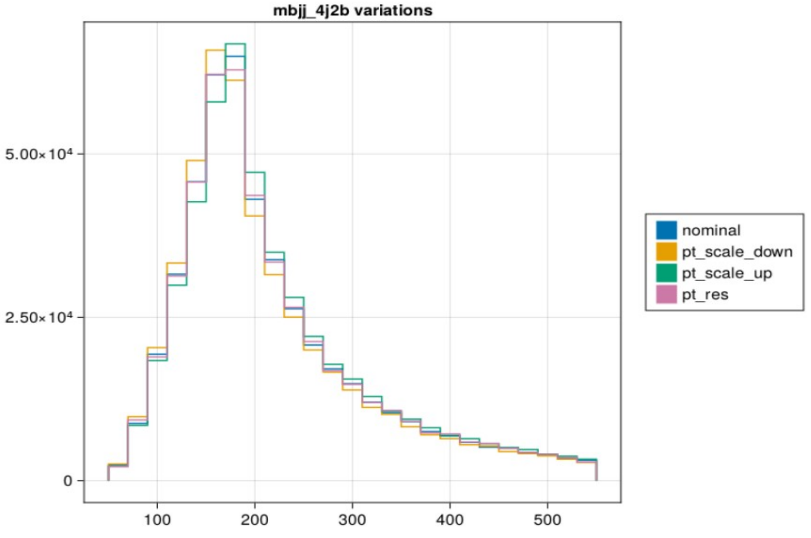
Results

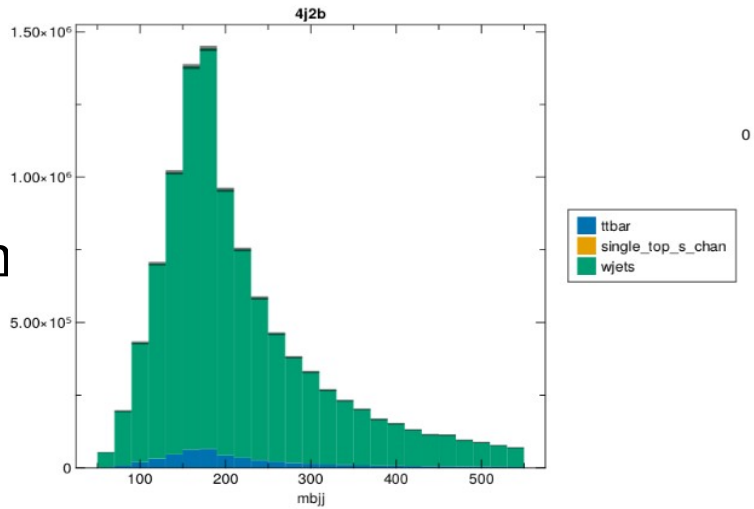
Distributed version



Results

- Convenient visualisation tools
- The workspace can be fully exported to a JSON file compatible with Cabinetry/Pyhf
- Found some issues in the reference implementation





What Have I Learned?

- First time working with distributed computing
- Some statistical insights about the inner workings of AGC
- Got more understanding of HEP