Julia in Trigger Level Analysis of $Z' \to b\bar{b}$

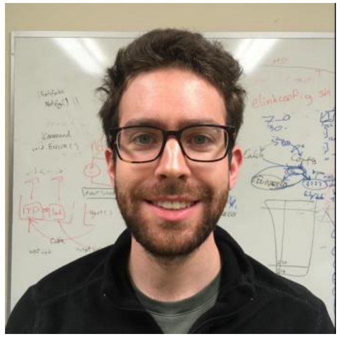
MICHAEL FARRINGTON

Analysis Team

Jerry Ling
PhD Student

Michael Farrington
PhD Student

Stefano Franchellucci
PhD Student

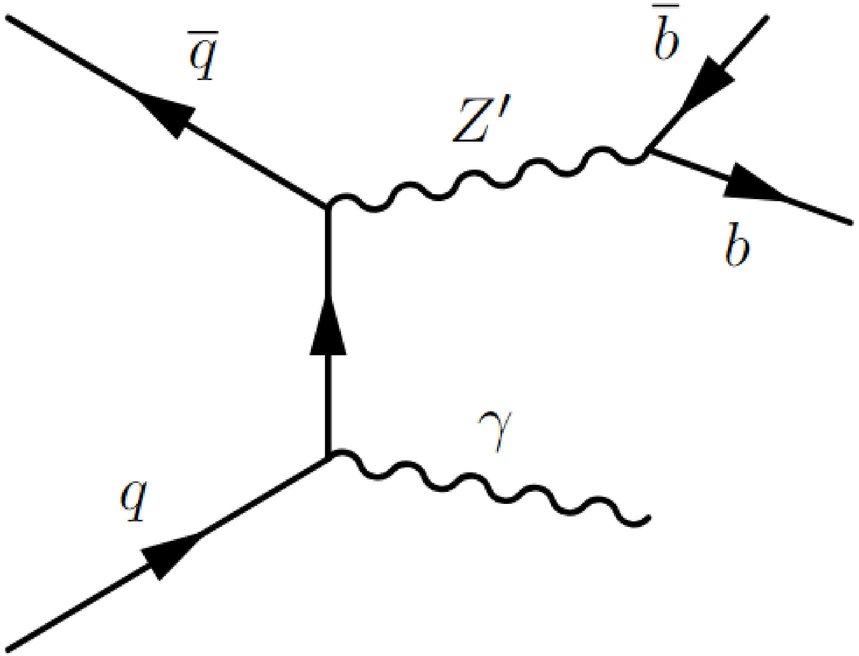


Aaron White
Postdoc

Rongkun Wang
Postdoc

Melissa Franklin
Professor

Physics Model



—We are searching for a new heavy, neutral vector boson Z'

—The Z' is produced in association with an initial state radiation (ISR) photon and decays to two bottom quarks

—The trigger requires 2 jets with $p_T > 25$ GeV and a photon with $p_T > 35$ GeV

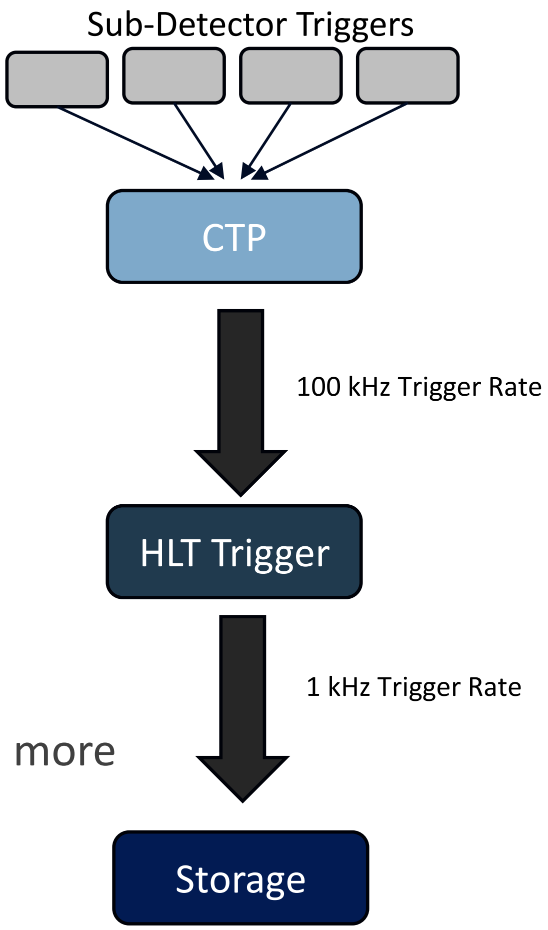
ATLAS Trigger Chain

—The ATLAS Trigger Chain can be divided into three main steps

1. **Sub- Detector Level Triggers:** Individual sub-detectors implement a trigger based on raw kinematics

2. **Central Trigger Processor (CTP):** The CTP collects and makes a trigger decision based on a combination of all the sub-detector triggers

3. **High Level Trigger (HLT):** Final trigger decision made at the software level with more accurately reconstructed variables

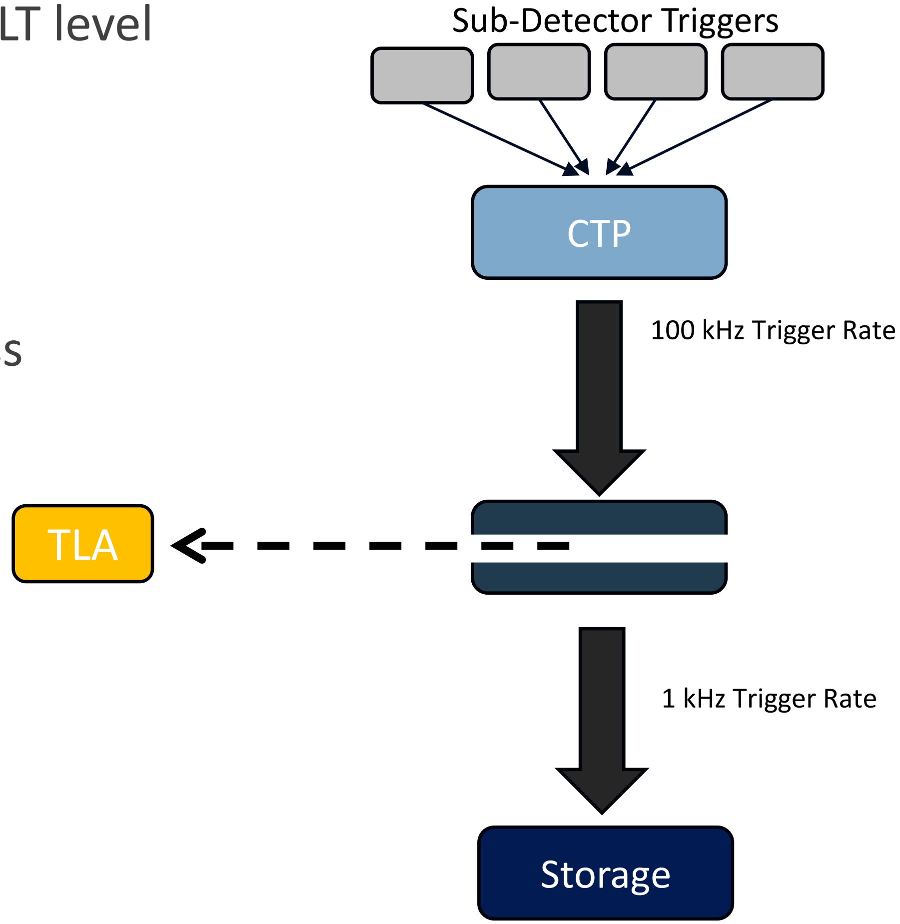


Trigger Level Analysis

- Trigger Level Analysis (TLA) let's us get in the middle of this chain by choosing to save partial event information at the HLT level to use a looser trigger

- We get a higher event rate and no pre-scaling, but get less information in each event e.g. no muons, no tracks

- This makes TLA competitive when you want to set low p_T triggers and are statistically limited

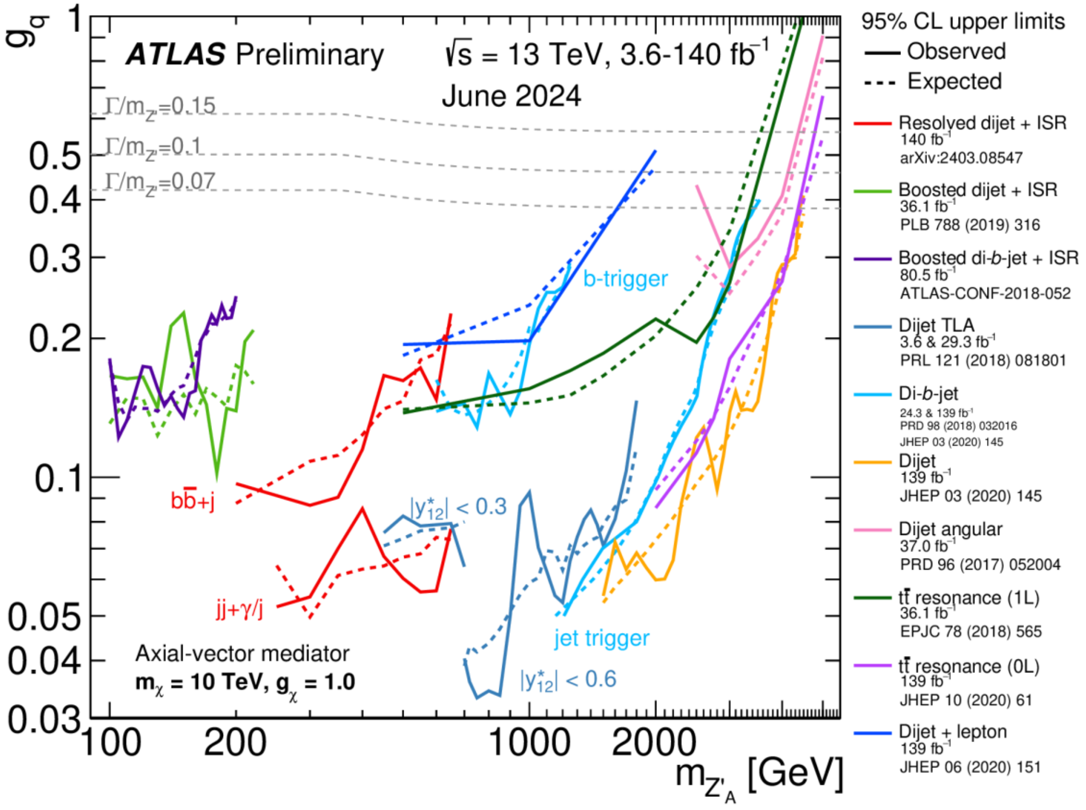


Current Z' Limits

—The goal is to constrain the coupling between the Z' and quarks denoted g_q

—The aim is to set limits in the sub 200 GeV mass range

—The goal is to set limits down to $g_q \leq 0.1$

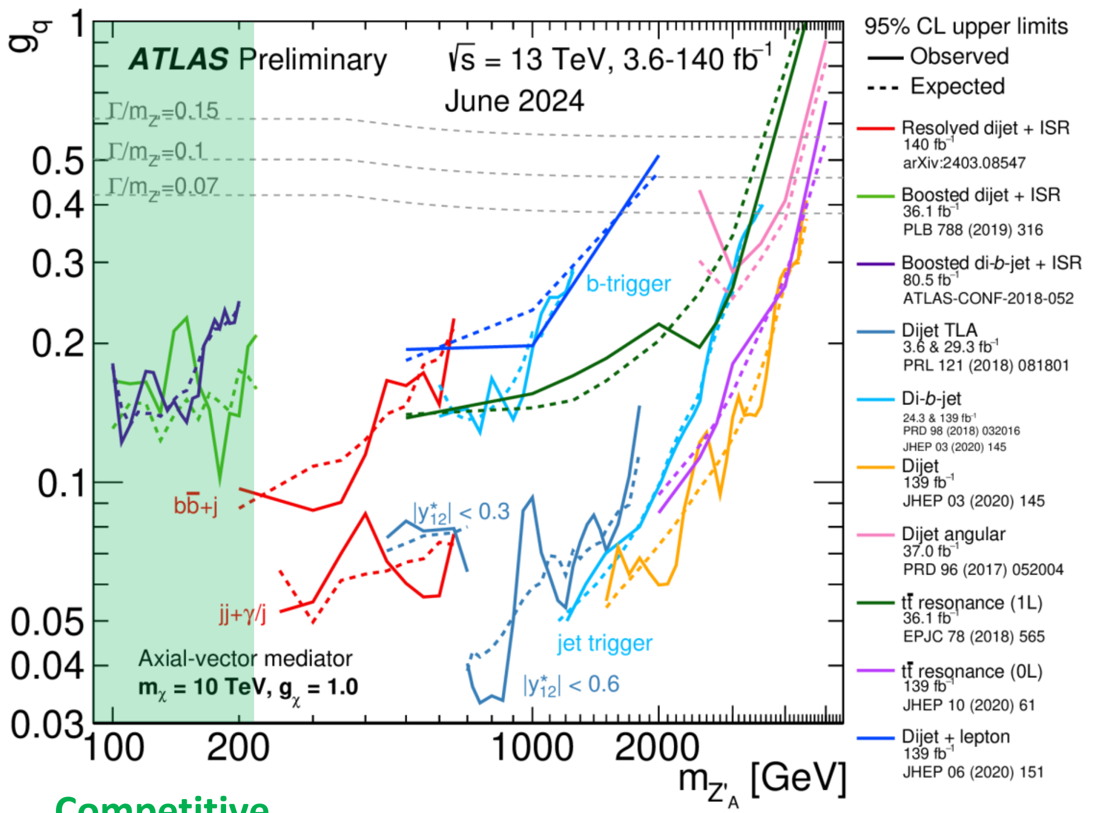


Current Z' Limits

—The goal is to constrain the coupling between the Z' and quarks denoted g_q

—The aim is to set limits in the sub 200 GeV mass range

—The goal is to set limits down to $g_q \leq 0.1$



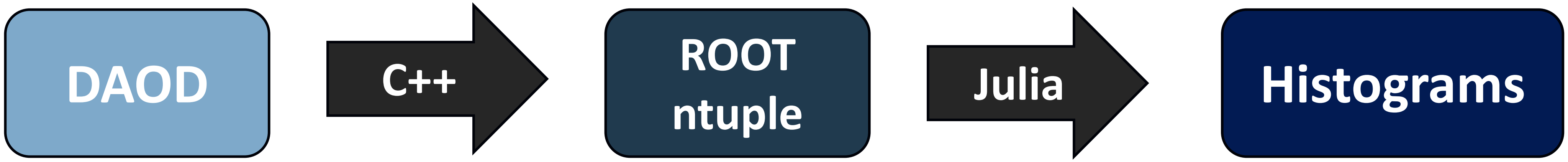
Competitive Region

B-Tagging in TLA

- To identify jets produced by bottom quarks in our detector, we use a deep sets algorithm that is fed track level variables in the HLT

- This is a standard ATLAS algorithm which is typically used in the HLT as a loose b cut

Workflow Overview



- The workflow starts by producing ROOT ntuples from the standard ATLAS data format (DAOD) using C++

- Once custom ntuples are made, Julia can be used to decode them and make and analyze histograms

- For this study we have both signal MC and a few runs of unblinded data to use

ntuple Production

—To produce ntuples, we need to use ATLAS software to loop through DAOD trees and decode custom objects where the data is stored

—Because the data is stored in ATLAS custom objects, this step of the analysis cannot be done in Julia

ntuple Reading

- The analysis ntuples are read out using UnROOT and are processed through our analysis cuts

- UnROOT LazyTree structure for reading out trees is fast and malleable, allows you to easily parse tree by rows or columns

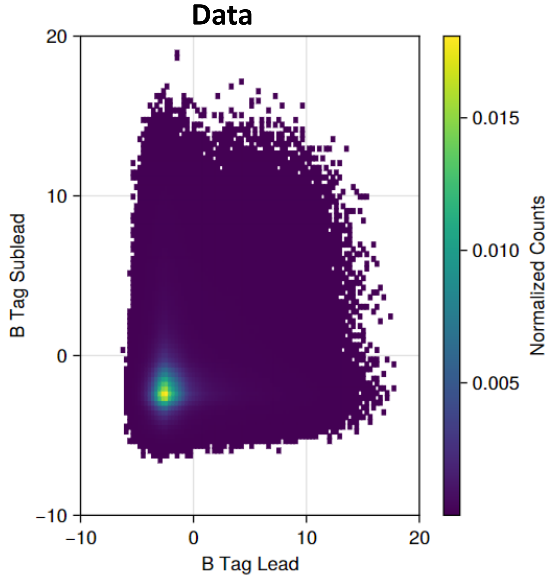
Cuts and Histogram Production

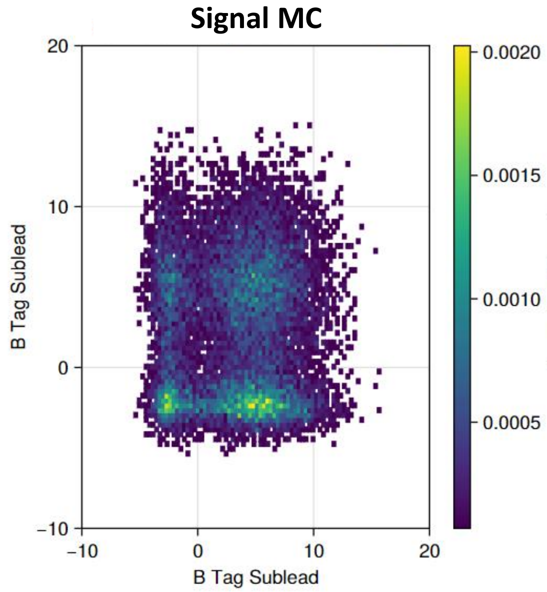
— The main source of background our cuts are trying to eliminate is non-resonant QCD background

— The primary cut is to require both the leading and subleading jets to be "b-tagged" which in this case means their b-score is above 0

— Additionally apply cuts requiring photon to be isolated and the two jets to be close in η i.e. $y^* < 0.8$

— Histograms are then made using FHist, very easy package to use, lets you manipulate histograms e.g. rebinning



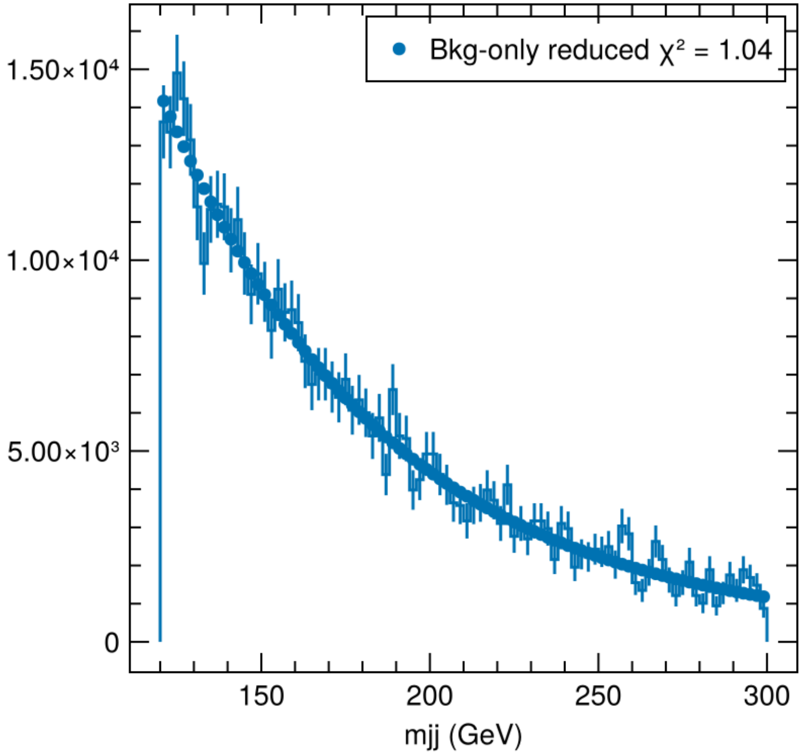


Mass Histogram

—The analysis centers around performing a "bump hunt" on the invariant mass distribution of the leading two b jets

—With the b jet mass histogram, a fit can be performed to find and subtract off the background

—Need to use PyHF for this step, there is currently no standard HEP Julia package for fitting



Histogram Fitting

— The analysis fits a falling distribution from the ATLAS recommendations to model the background as

$$f(x) = p_1(1-x)^{p_2} \; x^{-p_3+p_4 \log x}$$

where $x = m_{jj}/\sqrt{s}$ and the p_i are the parameters to be fitted

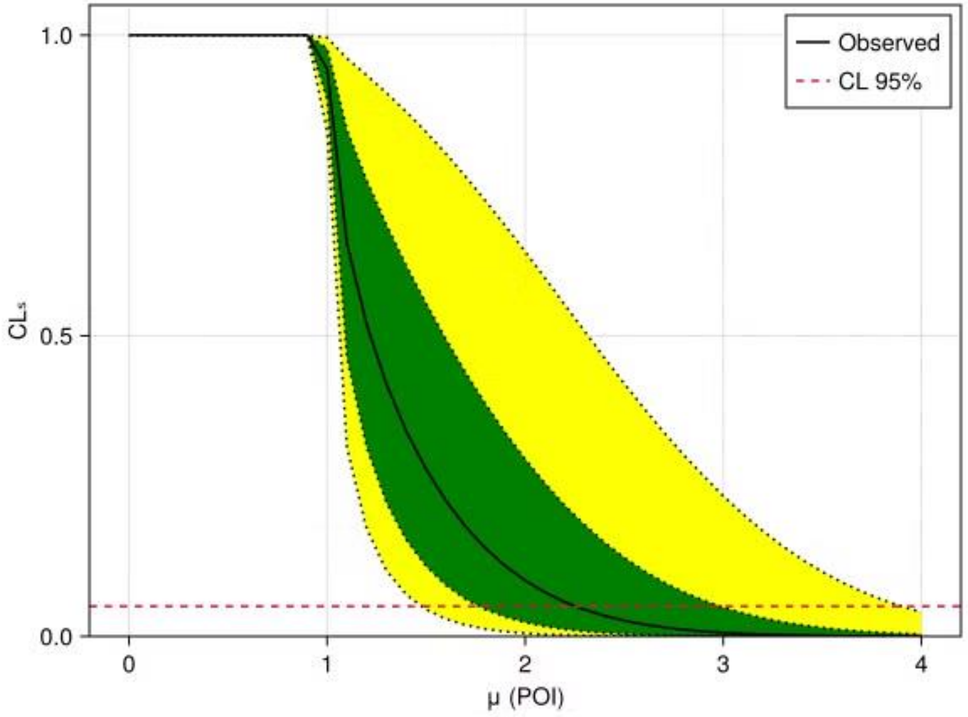
— Using a power law offers enough flexibility to fit our background while still being “rigid” enough to not overfit and eliminate any narrow resonance signals

Sensitivity Studies

— To optimize cuts, we use PyHF to set an upper limit on the signal strength μ or equivalently a lower limit on g_q

— PyHF is taking the falling function we fit to the mass distribution as the background model and the Z' MC as our signal model

— Then a likelihood method is used to estimate how much signal we could be seeing in our data



Julia in the Main Analysis

—Julia is extensively used in the main analysis workflow once ntuples have been produced

—Julia offers many useful tools for reading ROOT files, making histograms, more exploratory ML techniques, etc. so it’s what we use “day-to-day”

—There are still unfortunately non-Julia portions of this workflow in ntuple production and signal model fitting as well as in peripheral tasks e.g. calibrations, simulations, etc.

Hemisphere Mixing

—Attempt to use a data-driven method to model the QCD background in our data sample by creating synthetic data out of real events

—Premise relies on the fact that in a true Z' event, there should be delicate correlations in the kinematics that aren't present in background

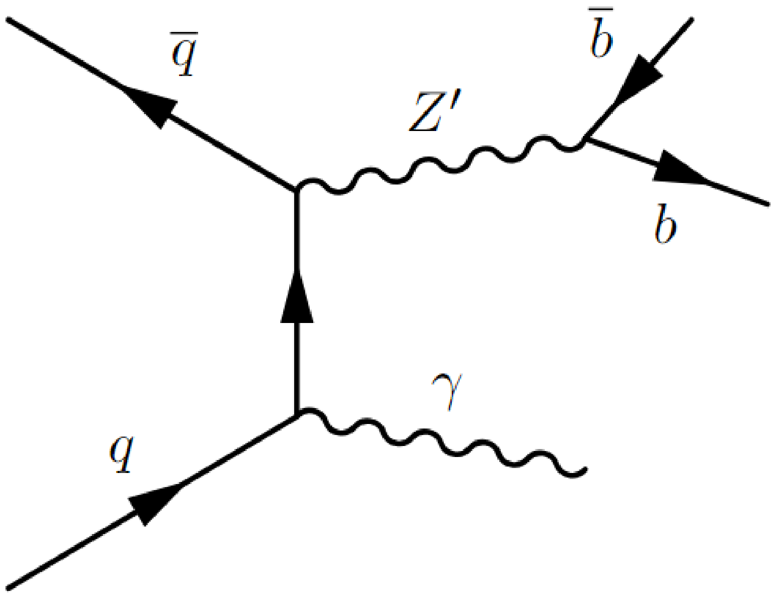
—Procedure

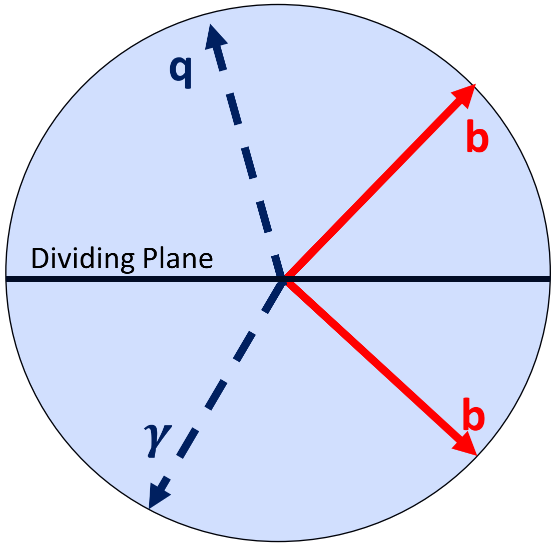
1. Split a data event geometrically into two hemispheres
2. Pair two hemispheres by minimizing a chosen metric, in this case $|E_{T1} - E_{T2}|$
3. Rotate the two hemispheres to match and form a new synthetic event

Split

— The events are split by bisecting the angle between the two b jets

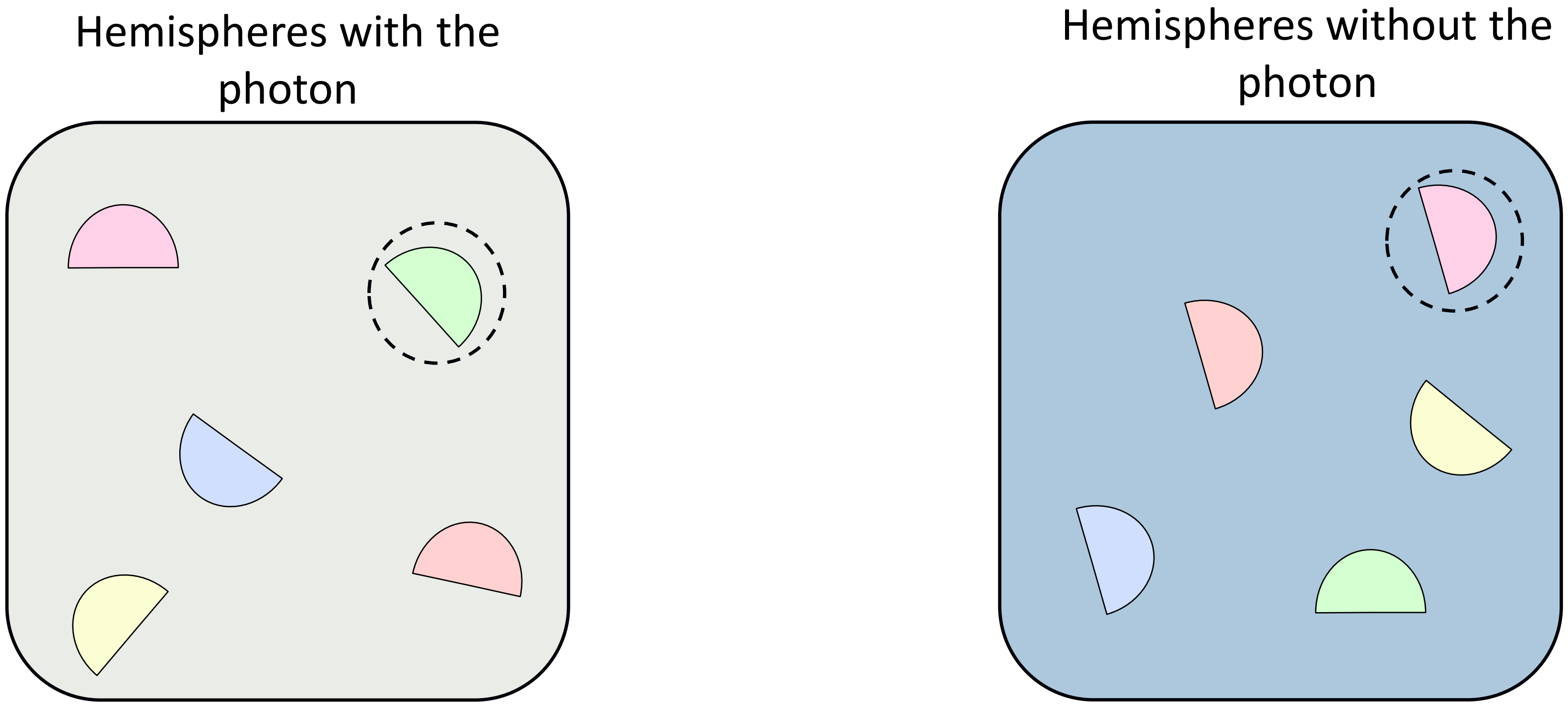
— Each hemisphere gets the jets/photons on its side of the bisecting plane





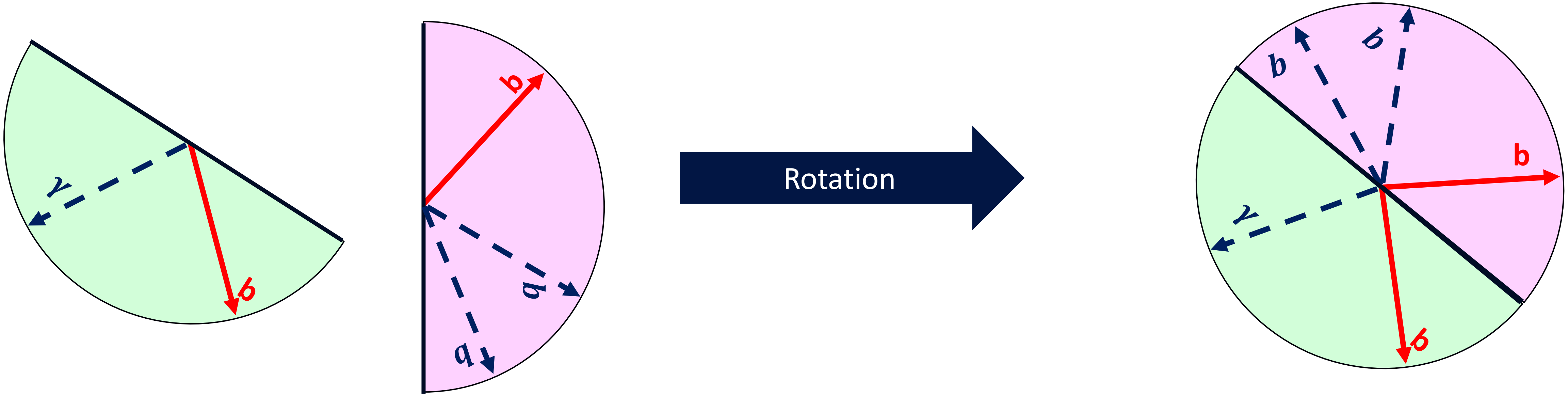
Pair

— Hemispheres a photon are paired with hemispheres without a photon based on our metric



Rotate

—The rotation is designed to make the planes splitting the hemispheres of two different planes overlap

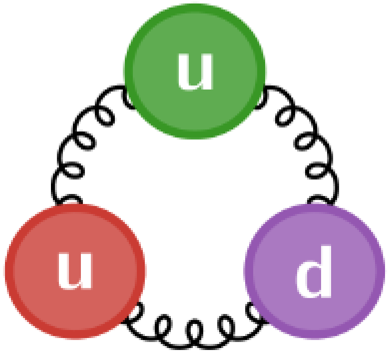


Julia in Hemisphere Mixing

— Julia offered tools which made development of this algorithm straightforward

— It was easy to convert from "ATLAS Coordinates" (p_T, η, ϕ, m) to Cartesian (E, p_x, p_y, p_z) using LorentzVectorHEP.jl

— Rotations.jl made it very easy to find the rotation matrices needed to match up hemispheres saves you from a lot of annoying geometry



Hemisphere Unit Tests

- Julia makes unit tests very easy to implement, incredibly helpful if you're learning the language

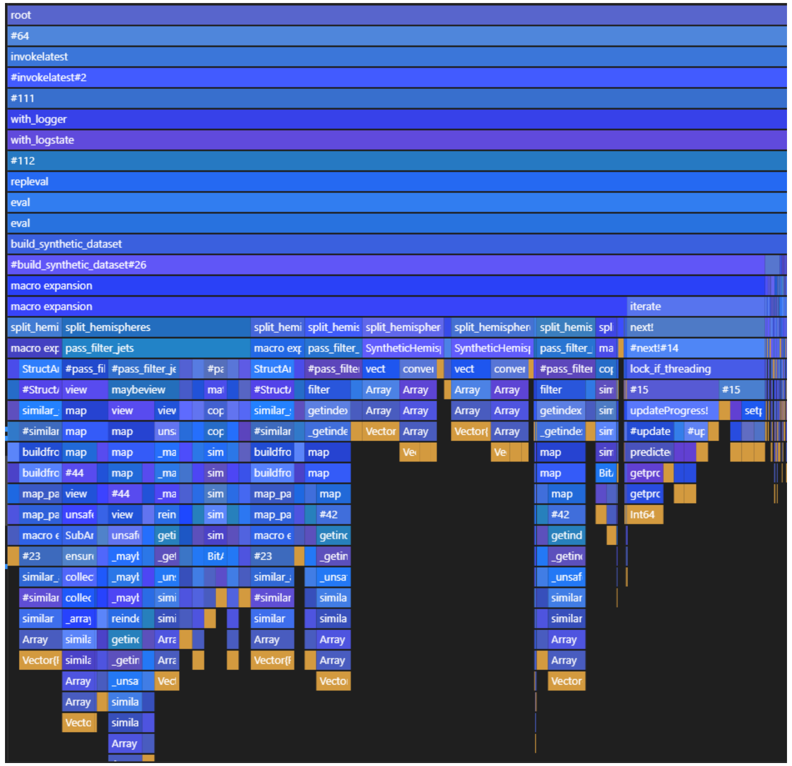
```julia
@testset "SyntheticDatasetUtils" begin
    for i in 1:100
        v1 = rand(3)
        v2 = rand(3)
        normalize!(v1)
        normalize!(v2)
        M = BjetTLA.antiparallel_rotation_matrix(v1, v2)

        @test dot(v2, M*v1) ≈ -1.0
        @test dot(BjetTLA.plane_bisector(v1, v2), BjetTLA.plane_bisector(v2, v1)) ≈ -1.0
        @test dot(BjetTLA.plane_bisector(v1,v2), v1) ≈ dot(BjetTLA.plane_bisector(v2, v1), v2)
    end
```

Hemisphere Profiling

- It was also incredibly helpful to have a profiler built in to optimize the hemisphere mixing algorithm

- It was even more convenient that this was built into VS Code as an extension!



Conclusion

- It is (mostly) possible to do a full ATLAS analysis using Julia

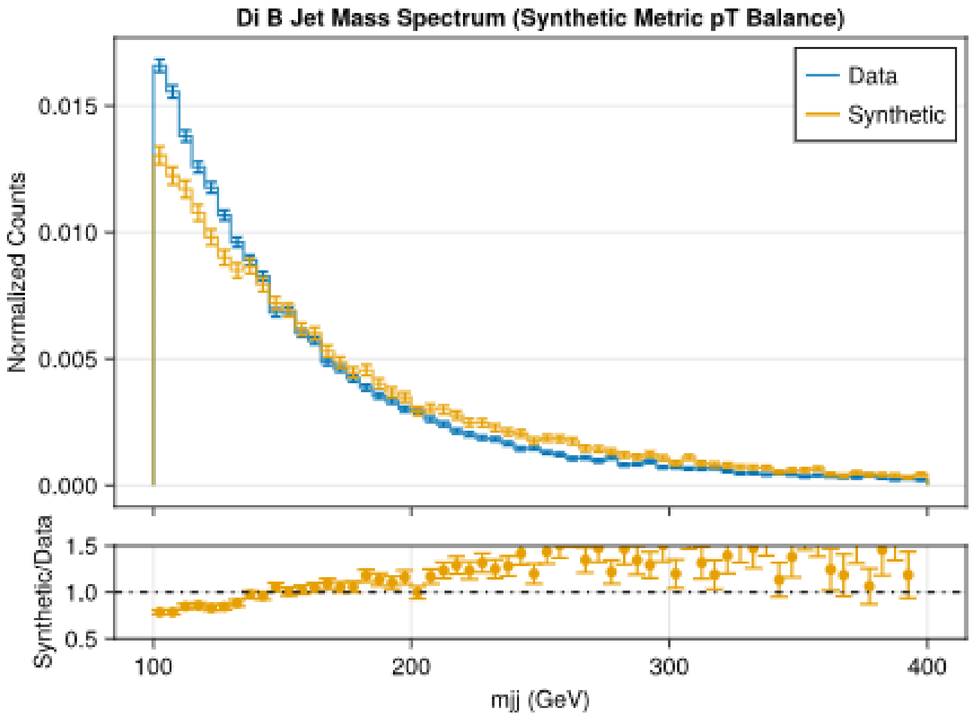
- There are a wide variety of tools available and they are easy to use for newcomers

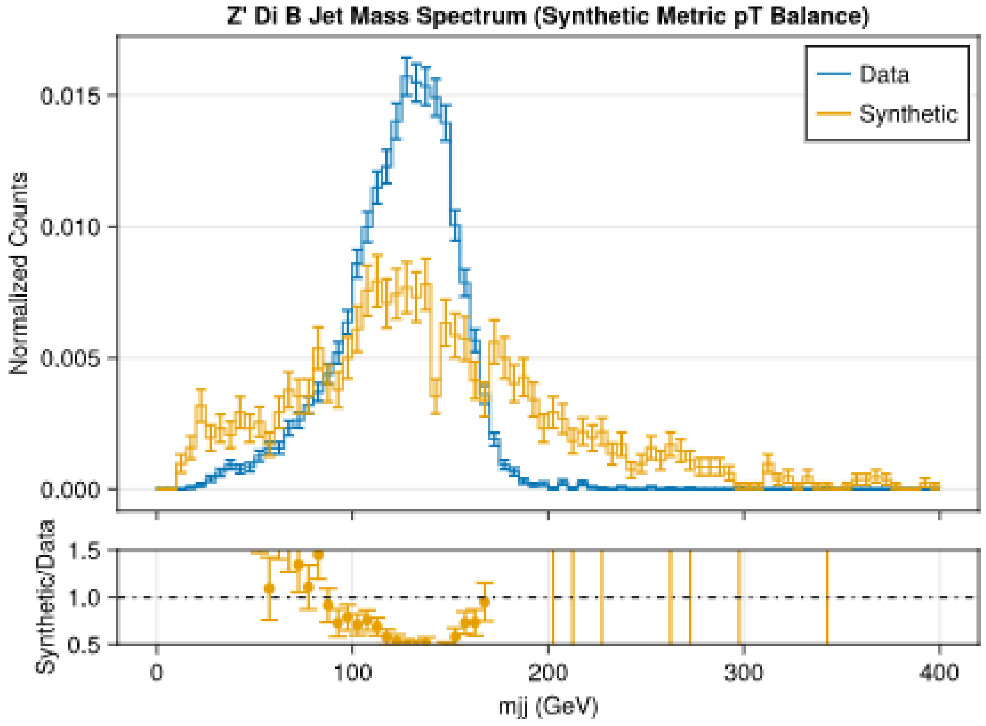
- Even a project like synthetic hemispheres which is a bit more niche than a "standard analysis" is easy to implement with packages

Questions

Backup

Synthetic Hemisphere Results





BDT for Cut Optimization

— To improve performance from our baseline cuts, we chose to switch to using a boosted decision tree (BDT) through XGBoost in Julia

— The BDT is currently trained on signal MC and data as background

— Challenge is to ensure that the BDT doesn’t sculpt a spurious signal into our mass distribution work is still ongoing to understand what variables are safe to feed to the BDT

fastDIPS Variables

— These are the track variables that are inputted into the fastDIPS algorithm

Input	Description
s_{d0}	d_0/σ_{d0}: Transverse IP significance
s_{z0}	$z_0 \sin\theta/\sigma_{z0 \sin\theta}$: Longitudinal IP significance
$\log p_{\mathrm{T}}^{frac}$	$\log p_{\mathrm{T}}^{track}/p_{\mathrm{T}}^{jet}$: Logarithm of fraction of the jet p_{T} carried by the track
$\log \Delta R$	Logarithm of opening angle between the track and the jet axis
IBL hits	Number of hits in the IBL: could be { 0, 1, or 2 }
PIX1 hits	Number of hits in the next-to-innermost pixel layer: could be { 0, 1, or 2 }
shared IBL hits	Number of shared hits in the IBL
split IBL hits	Number of split hits in the IBL
nPixHits	Combined number of hits in the pixel layers
shared pixel hits	Number of shared hits in the pixel layers
split pixel hits	Number of split hits in the pixel layers
nSCTHits	Combined number of hits in the SCT layers
shared SCT hits	Number of shared hits in the SCT layers

Table 1: Track features used as inputs for RNNIP and DIPS algorithms.