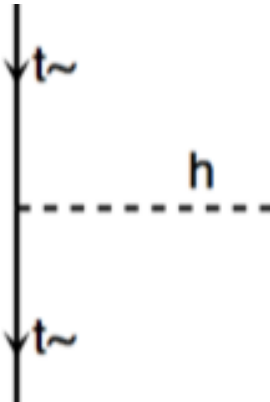
Using Foam as a Matrix Element Method Look-Up Table

Tom Sandell
April 18, 2017
University of Michigan
Advisor: Tancredi Carli
Working with Alexander Held

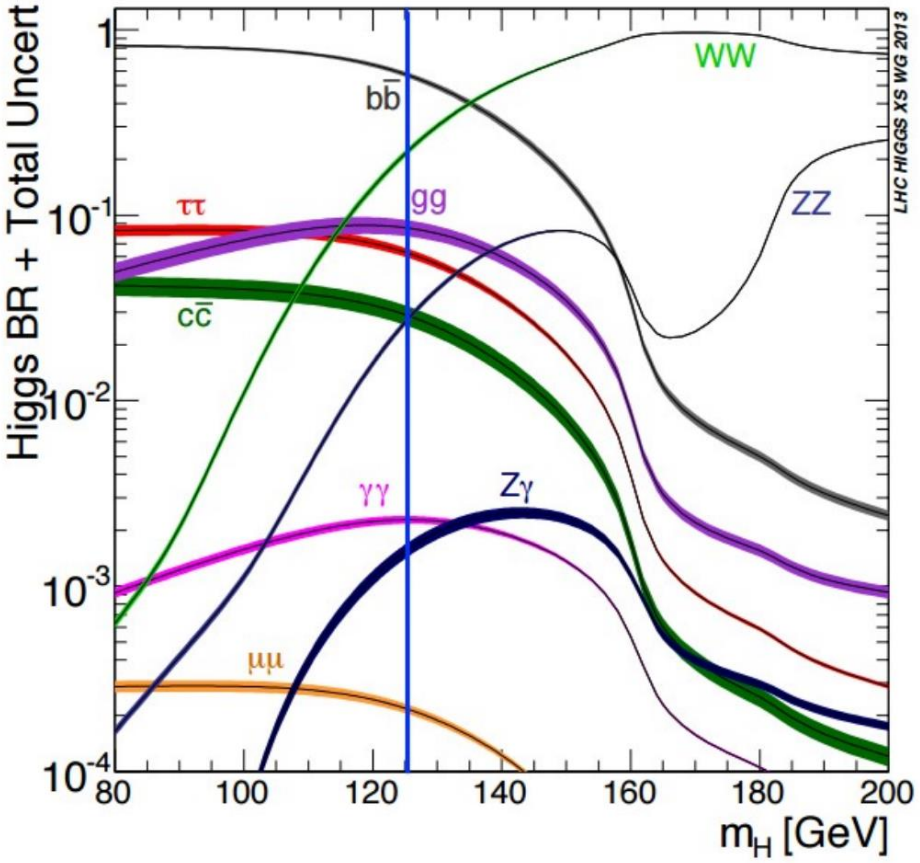
ATLAS Experiment

- Now that we found the Higgs Boson, we want to use it to test the Standard Model
- A good way to test this is with the Top Quark, because its large mass makes it sensitive to BSM effects
- We would like to test the top quark Yukawa coupling using the Higgs Boson coupling to top quark, to test for these BSM effects



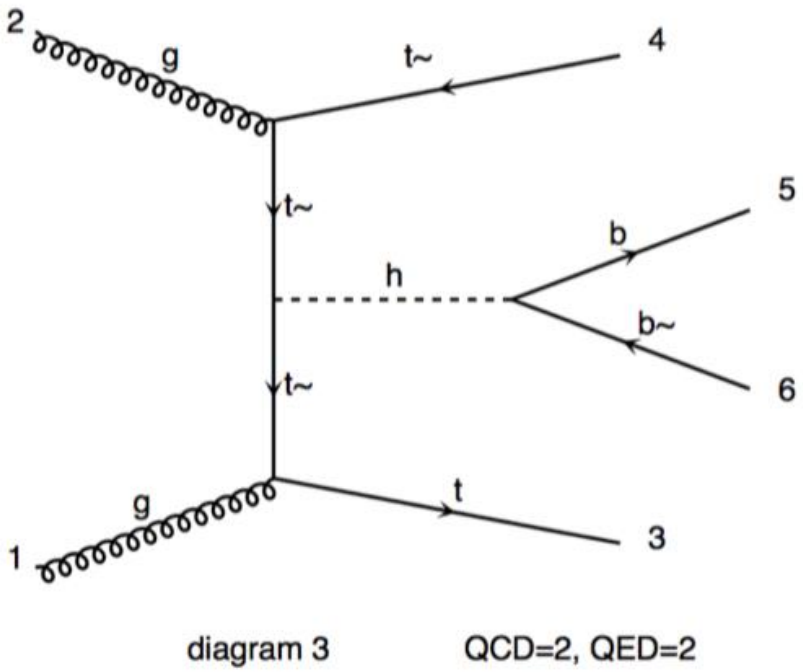
Higgs Boson Coupling to Top Quark

- There are many different decay processes that can measure this coupling
- There is a different analysis testing the coupling for each feasible decay process, after which each analysis will compare their results to find the most accurate value possible



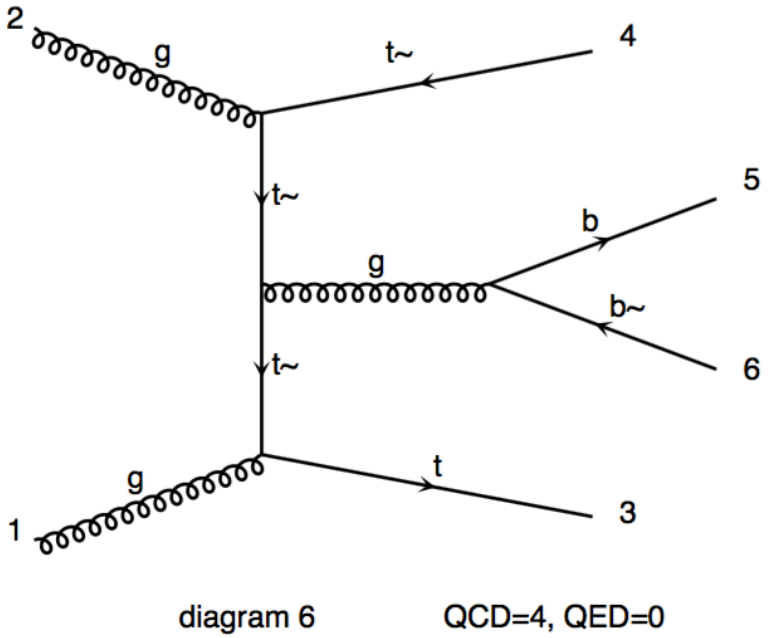
ttH(bb) analysis

- We focused on this particular decay process, where the Higgs boson decays to a bottom quark and an anti-bottom quark
- This is the most common decay process from which we can measure the coupling



Issue with ttH(bb) analysis

- There is a common background decay process (ttbar) that results in the exact same decay products to ttH(bb)
- Without a feasible and accurate way to discriminate between these processes, the uncertainty is too high on the value of the coupling to reach any definitive conclusions

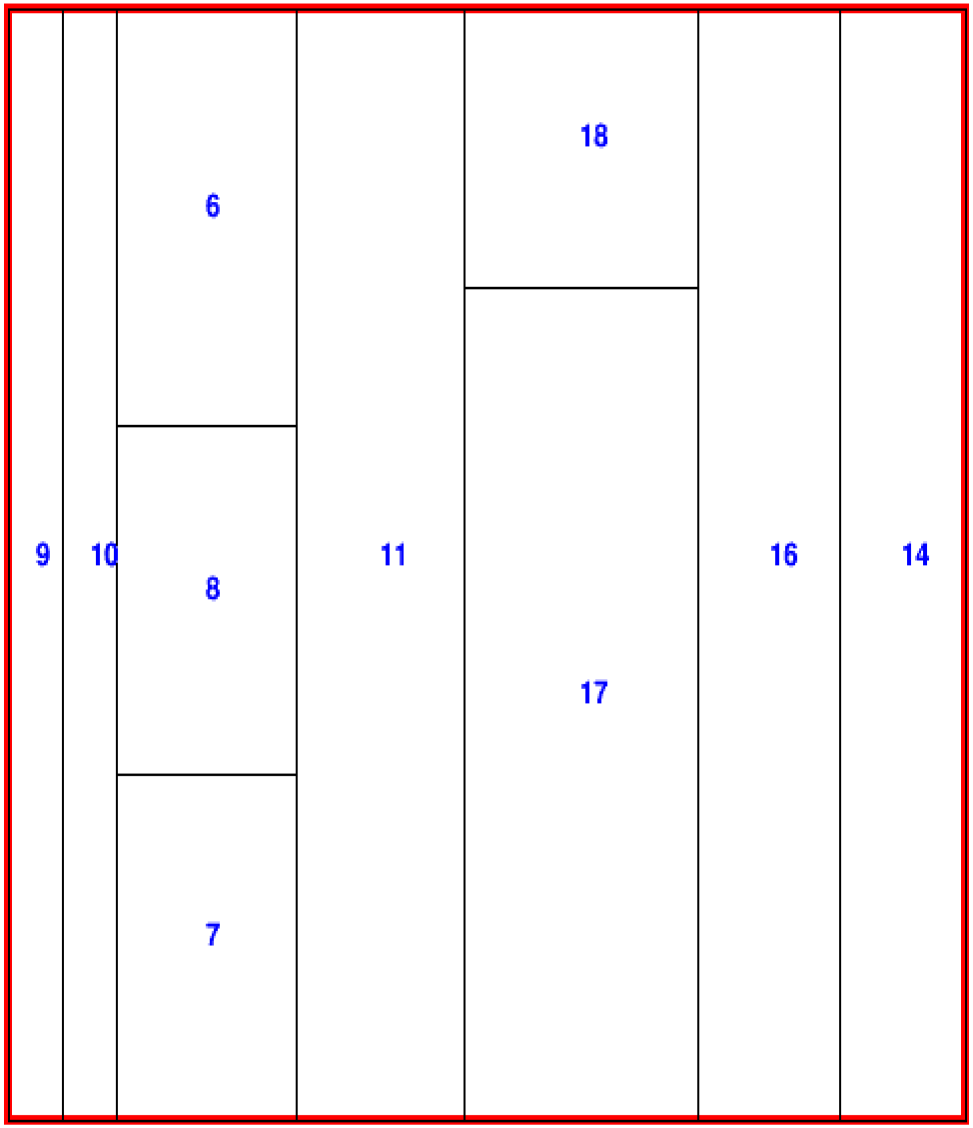


Matrix Element Method

- One bottleneck in this process involves the matrix element method, which attempts to extract theoretical information from physical events
- This method combines theory and physical events to accurately determine physical parameters based on complex systems
- The value that is calculated is the likelihood that the event is consistent with our hypothesis, i.e. that the event was a ttH(bb) decay.
- From this value, we can assume which process our event undertook by calculating a ratio of likelihoods (ttH(bb) / ttbar, for instance)
- The main issue with this method is calculation times

Foam Method

- Instead of calculating the matrix element for every event, we store this on a look-up table of n dimensions, where n is the number of parameters
- Our look-up table consists of an n-dimensional data structure with non-equidistant binning, called foam
- This is basically an n-dimensional space consisting of hyperrectangles, inside each of which an integral is calculated referring to the probability of your hypothesis

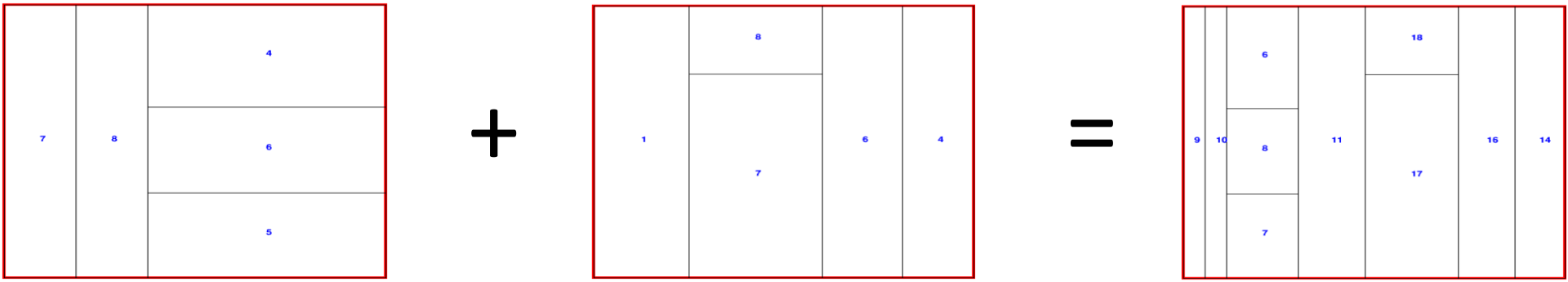


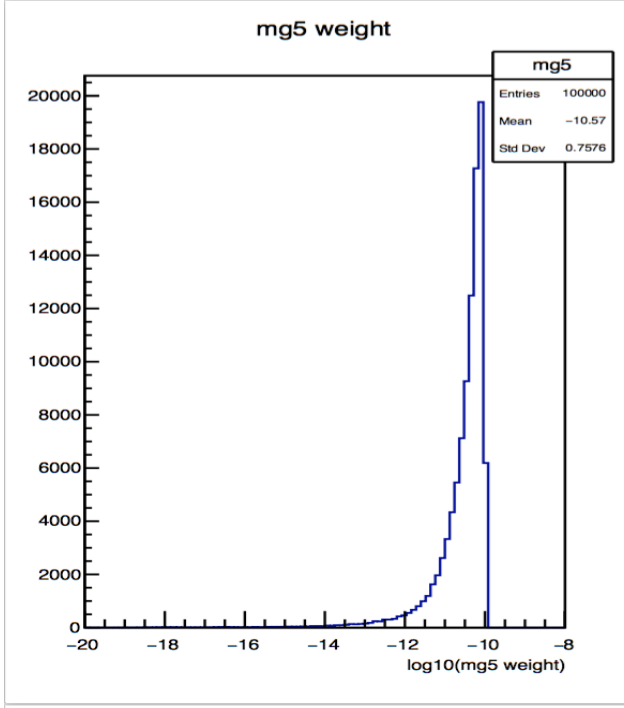
Foam Method

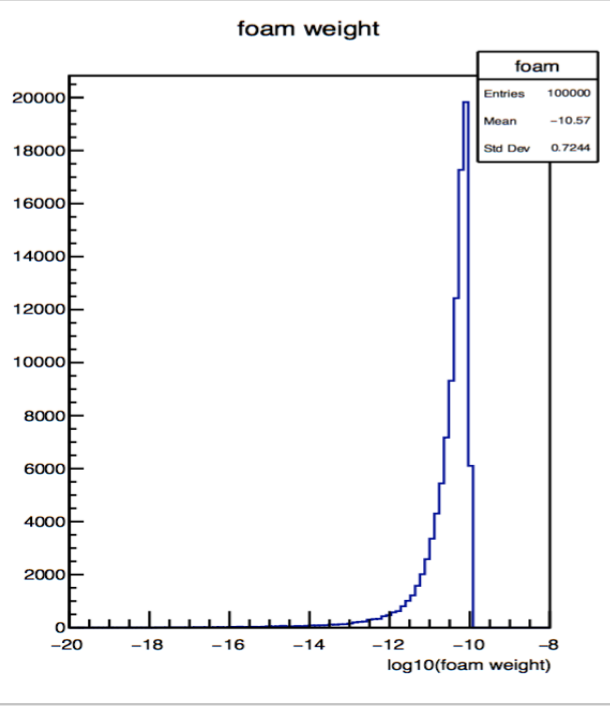
- The likelihood calculated by the foam method becomes much more accurate when many cells are used
- Unfortunately, it also becomes very slow
- My overarching project was finding ways to optimize foam so the buildup is fast enough that it is a feasible alternative to the current matrix element implementation
- This also attempted to confirm whether using foam as a matrix element look up table is feasible at all
- My project included efforts to create more cells in a foam and to make the calculated integral of existing cells more accurate

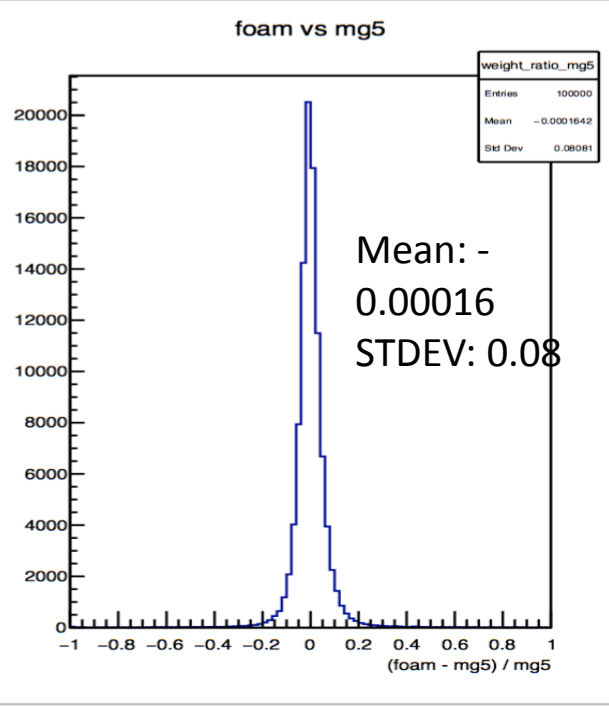
Foam Optimization

- My first project was to optimize foam by running multiple processes in parallel
- I did this by creating many small “subfoams” with splits defined by an especially accurate version of the foam algorithm, then combining them into one “master foam”
- This allowed us to increase the number of cells we could easily create by a factor of 64, from 100,000 to 6,400,000
- This resulted in a more accurate integral and thus a more accurate likelihood, which can be seen on the next slide

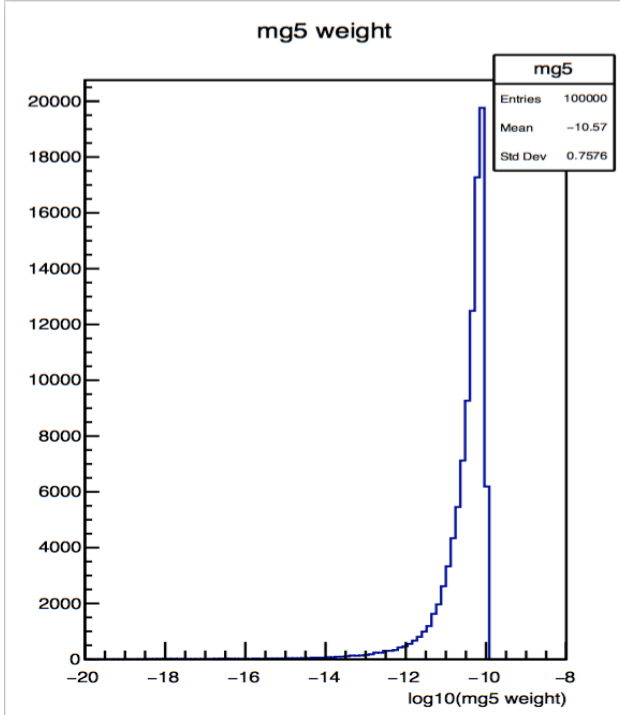


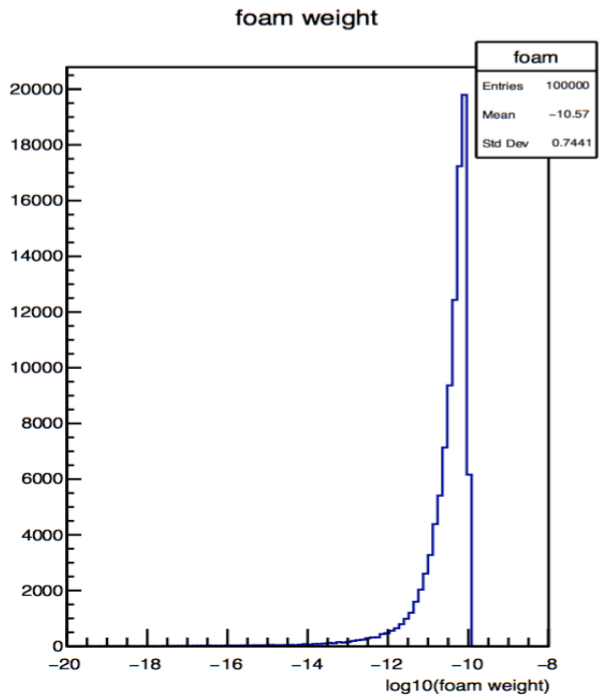


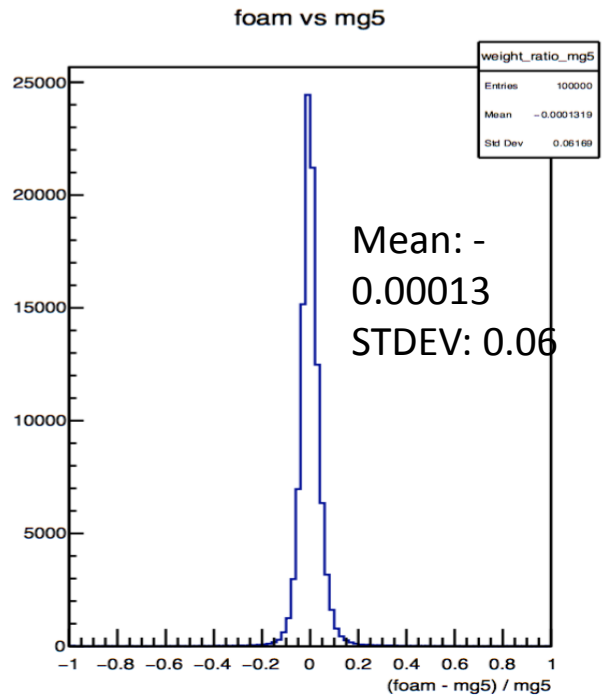




Mean: -0.00016
STDEV: 0.08





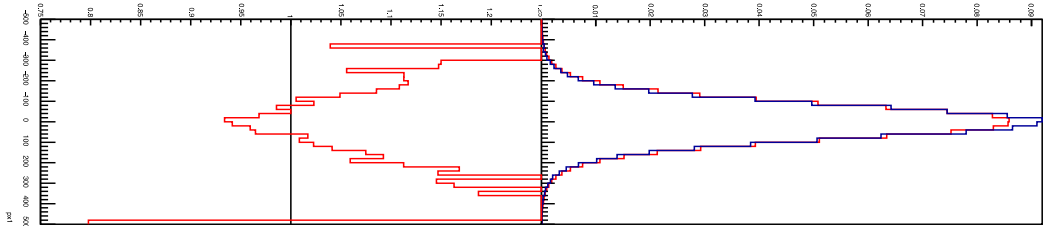


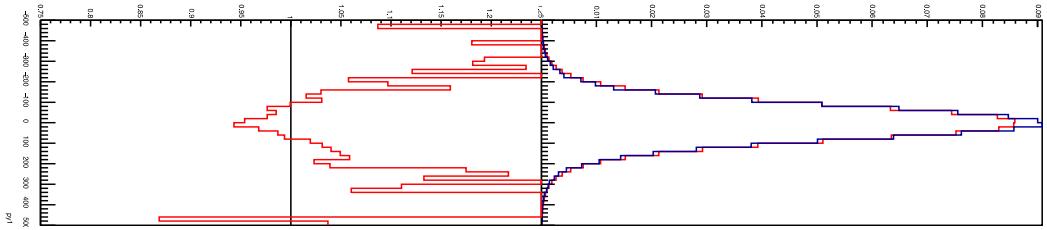
Mean: -0.00013
STDEV: 0.06

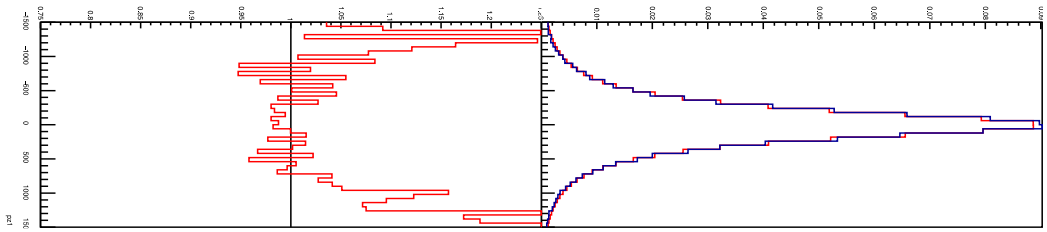
Foam Optimization

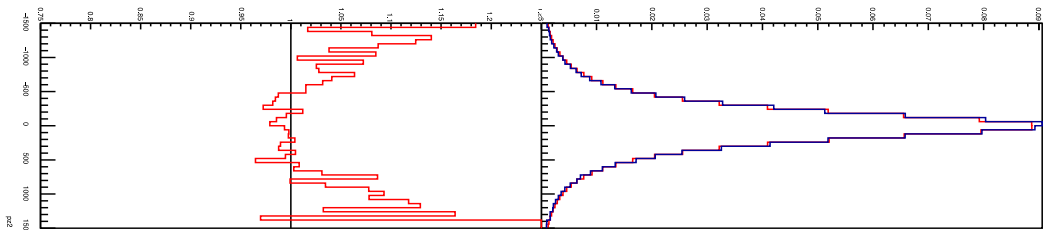
- Next, we focused on making the integral within each cell more accurate
- Foam has difficulty modeling strongly peaked distributions. This is because there may never be a hyperrectangle small enough to measure the very top of a peak
- So we focused on transforming variables, to eliminate strongly peaked variables from our distribution

100000 Cell Foam Modeling ttbar collision (px1, py1, pz1, pz2)









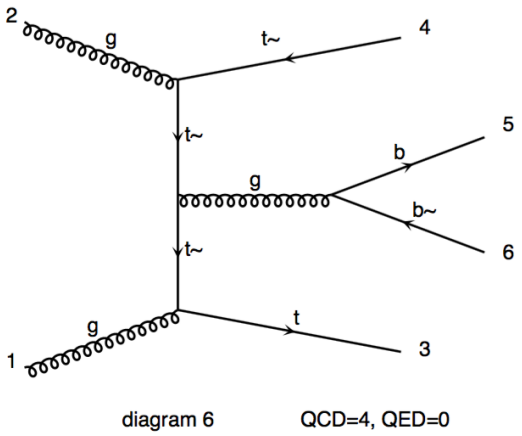
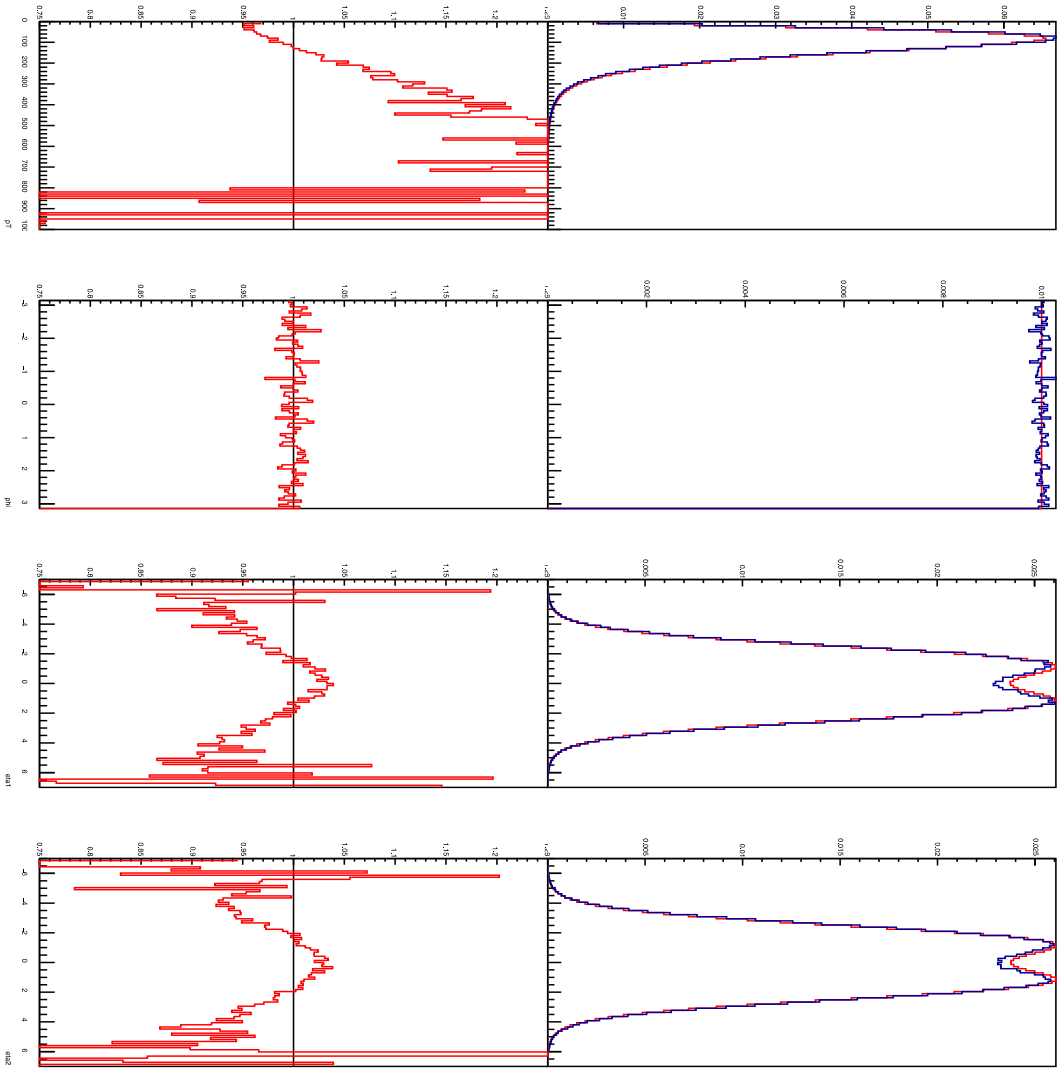


diagram 6 QCD=4, QED=0

100000 Cell Foam Modeling ttbar collision (pT, phi, eta1, eta2)



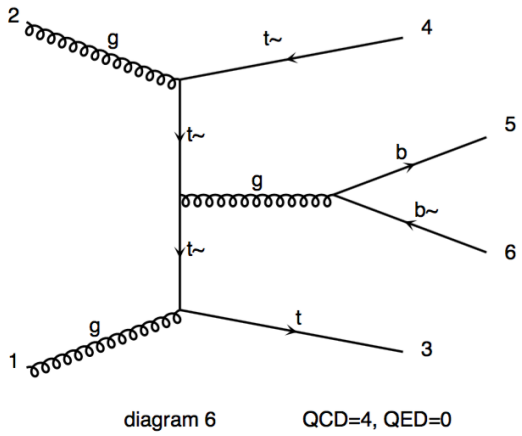
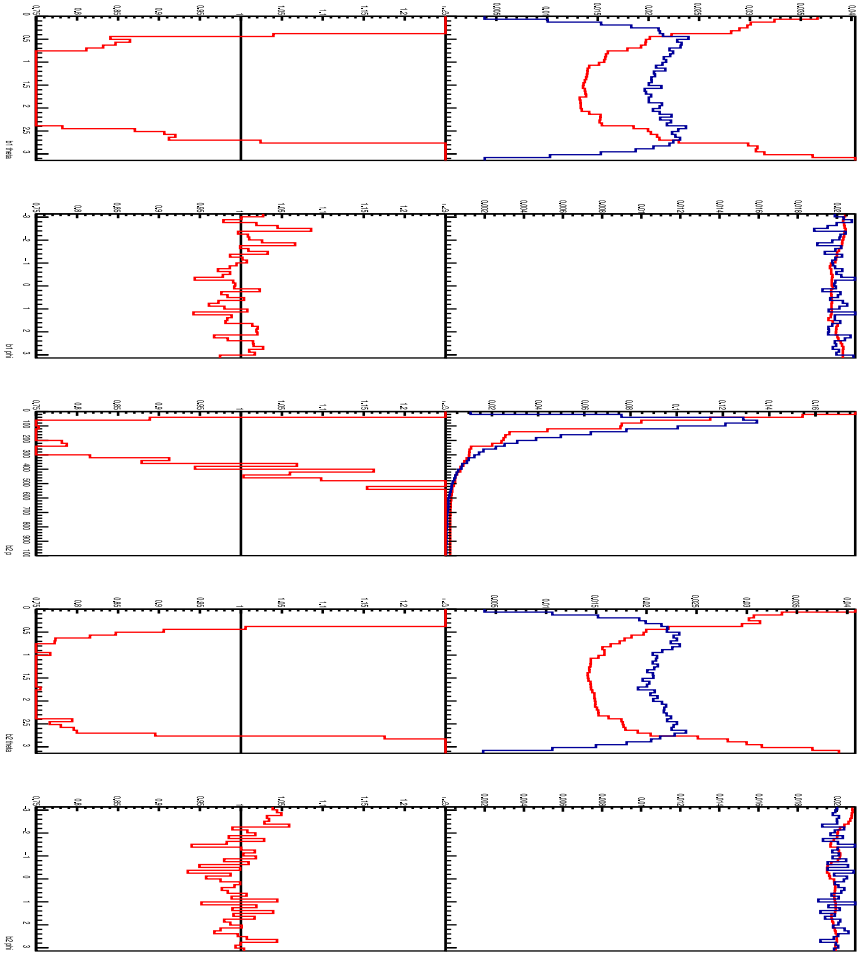
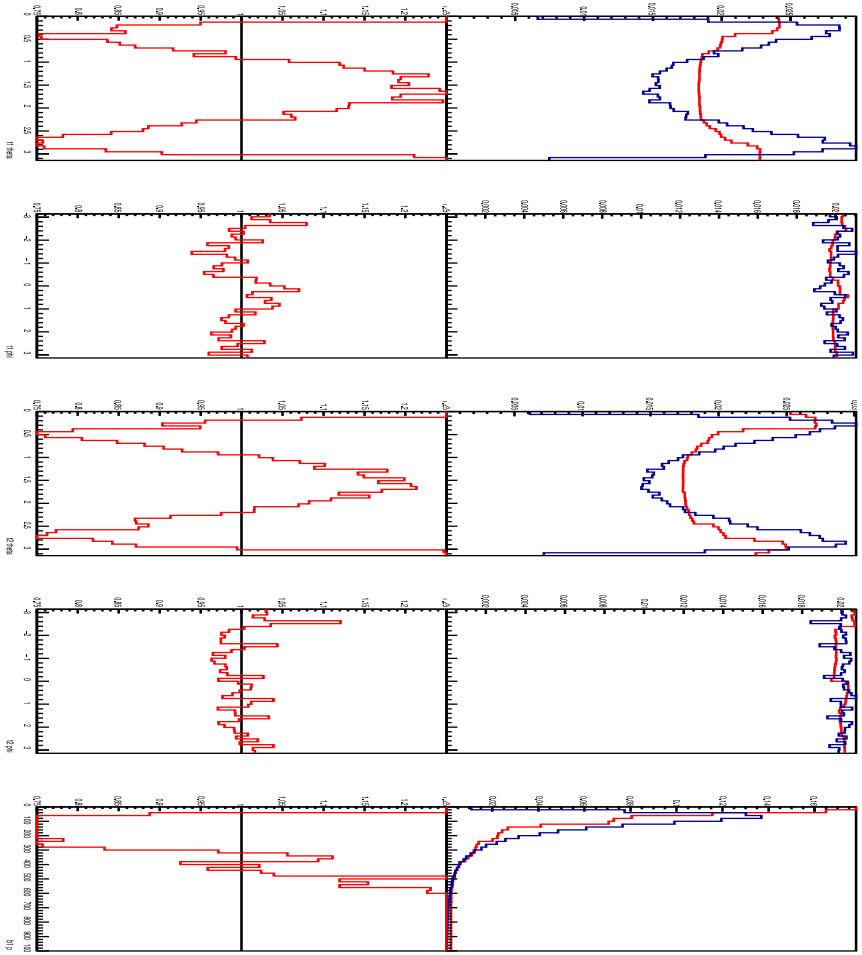


diagram 6 QCD=4, QED=0

Foam Optimization

- This worked well enough for a 4-dimensional foam, but after adding more decay products and working with a 10-dimensional foam, we needed to find better variables
- There is also a math issue with 10-dimensions which is hurting our fits that we still need to work through





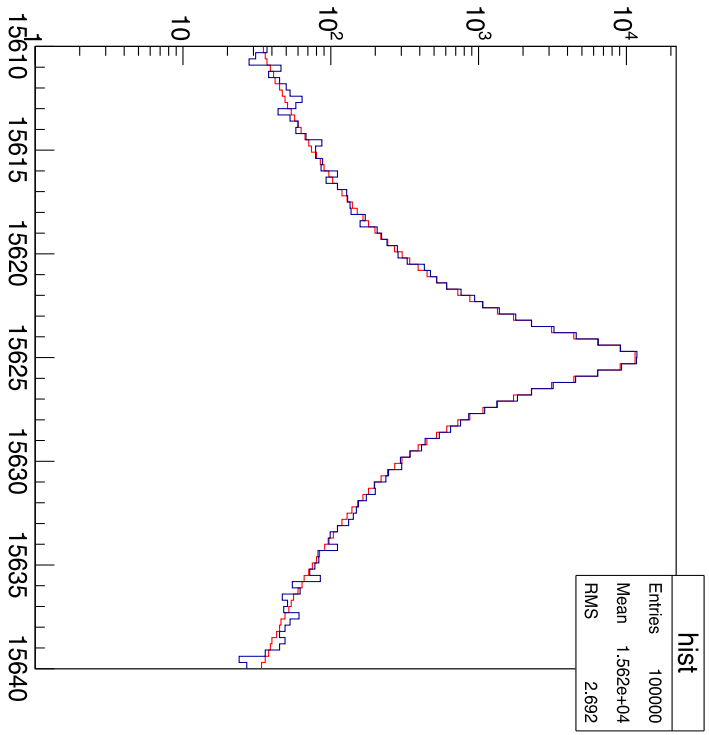
Foam Optimization

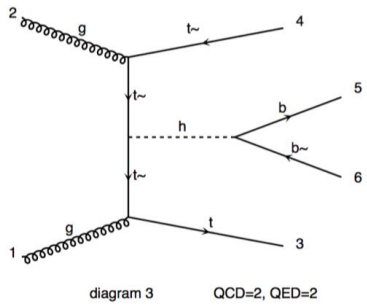
- Another variable transformation exists to flatten the transverse momentum variable for the foam, by converting it to the invariant mass distribution
- This works especially well for Higgs events, which have a strongly peaked and mostly symmetric angular momentum component
- As can be seen in the equation below, this changes the distribution into a constant based on the mass and width of the distribution
- The following distributions are from ttH(bb) events, not background events, so the distributions will look different than previous ones in my presentation

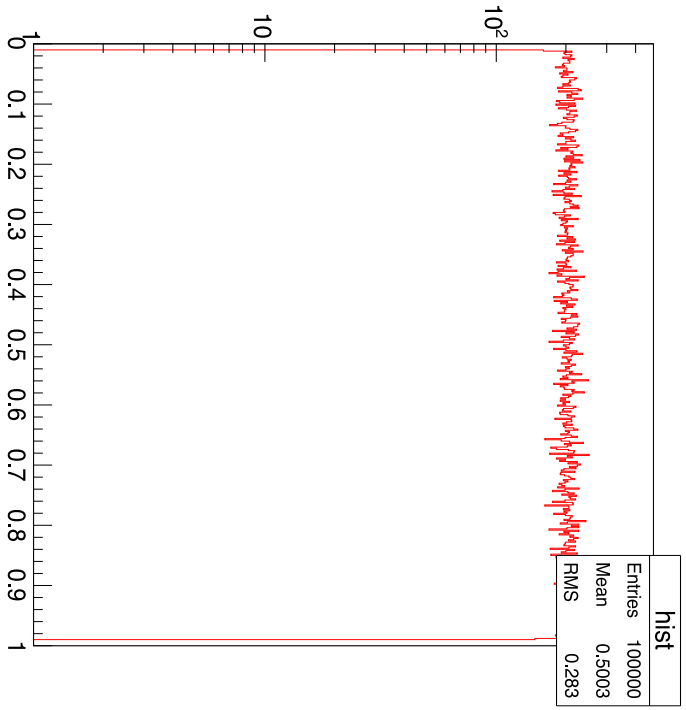
$$f(s_X)\,\mathrm{d}s_X \to f(s_X(y))\,\frac{\mathrm{d}s_X}{\mathrm{d}y}\,\mathrm{d}y = \frac{1}{\tan^2 y + 1}\,\frac{M_X\Gamma_X}{\cos^2 y}\,\mathrm{d}y = M_X\Gamma_X\,\mathrm{d}y$$

Variable Transformation

- On the left is the distribution of transverse momentum for ttH(bb) events, both using simulated events and by mapping our new variables onto the old ones as a proof of concept
- On the right is our new invariant mass variable, after these variable transformations. As can be seen, it is essentially perfect (completely flat)



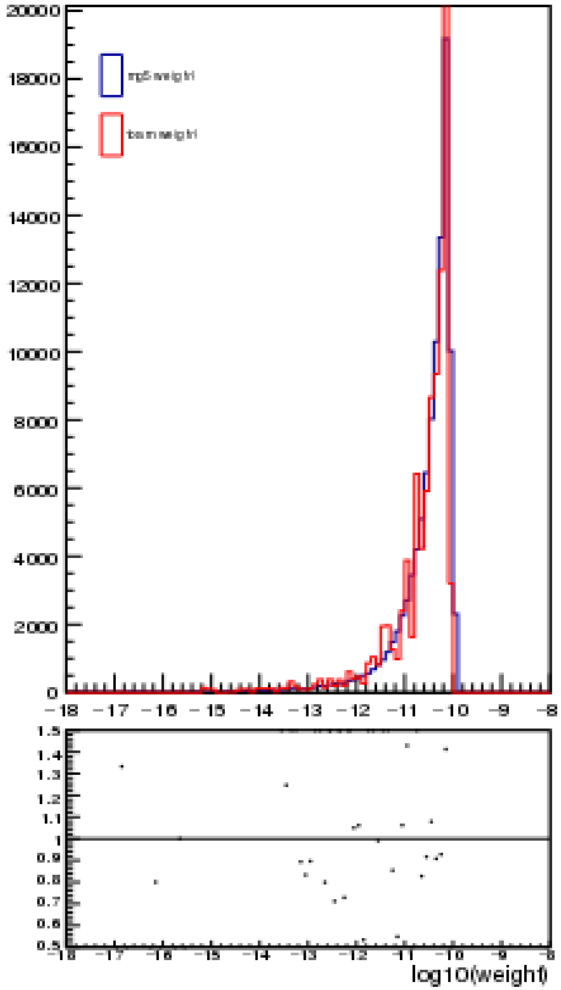




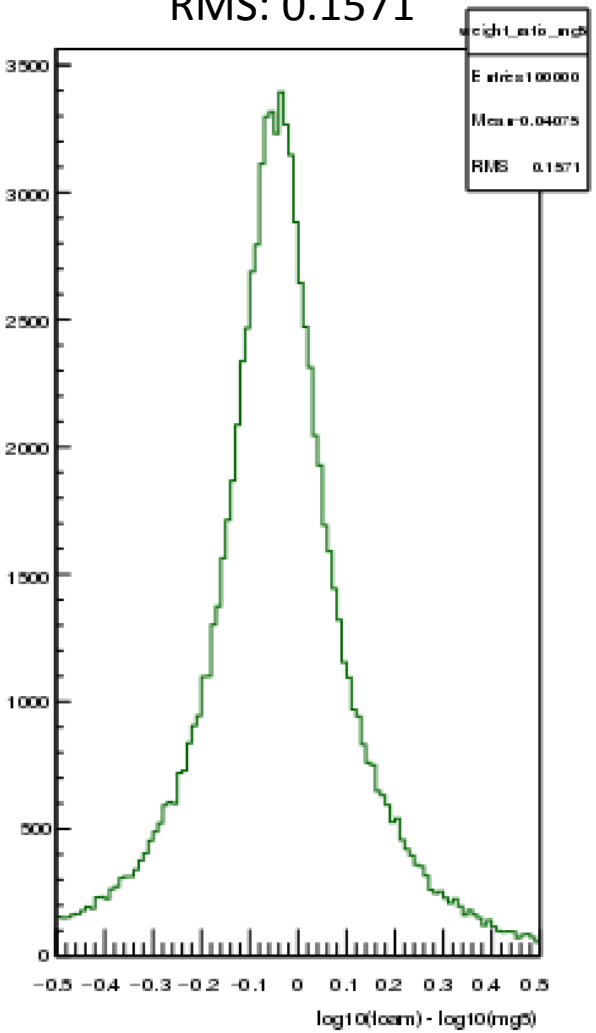
Foam Optimization

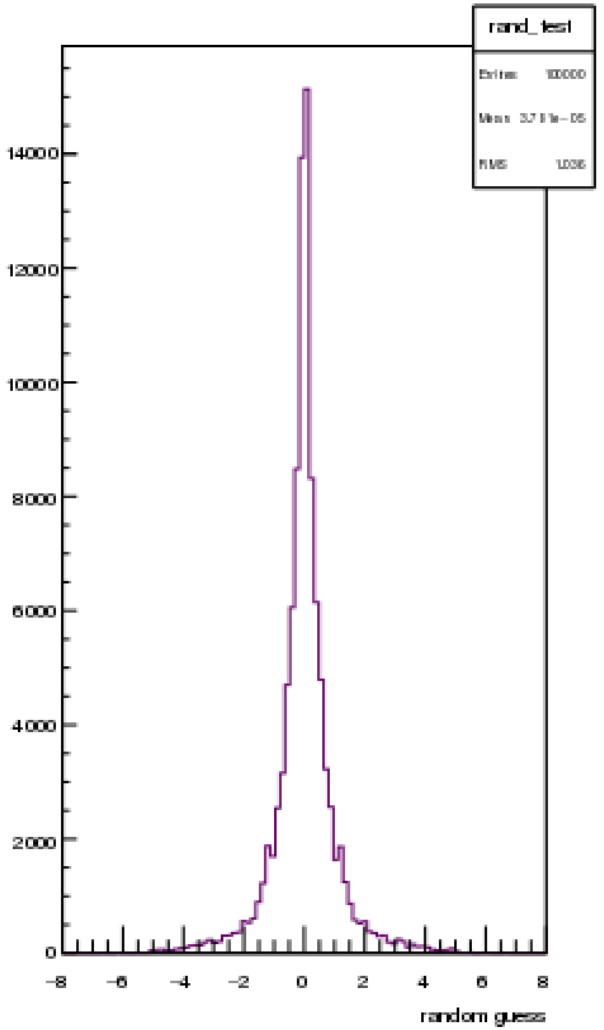
- Our next optimization involved implementing a “kernel” function, which samples neighboring cells along with the one an event is found in, to generate a more accurate integral and thus a more accurate likelihood
- This works especially well with distributions without many cells, as events on the border of two cells now weigh the integral calculated in both the cell they are in and the neighboring one

Foam Weight Comparison: No Kernel



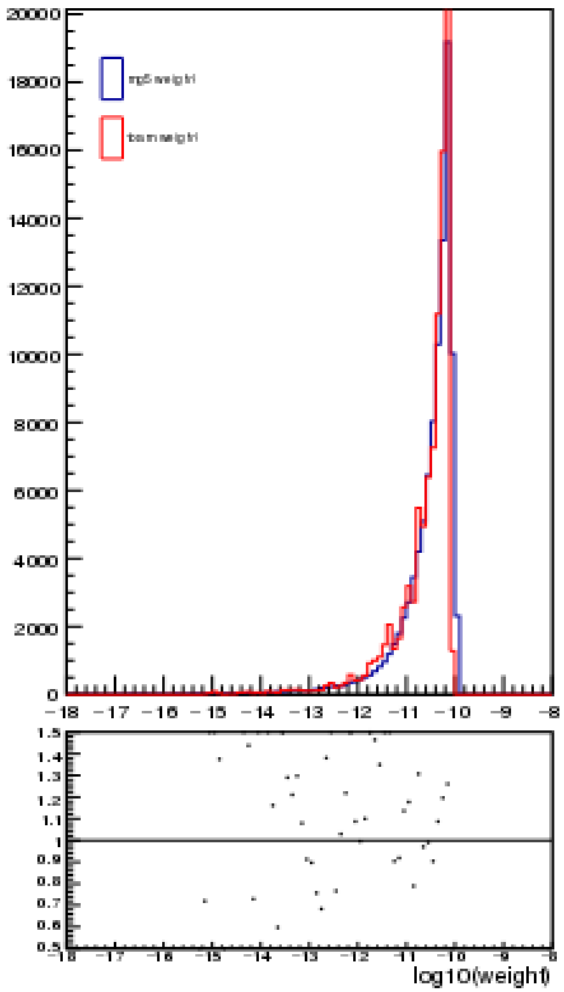
RMS: 0.1571

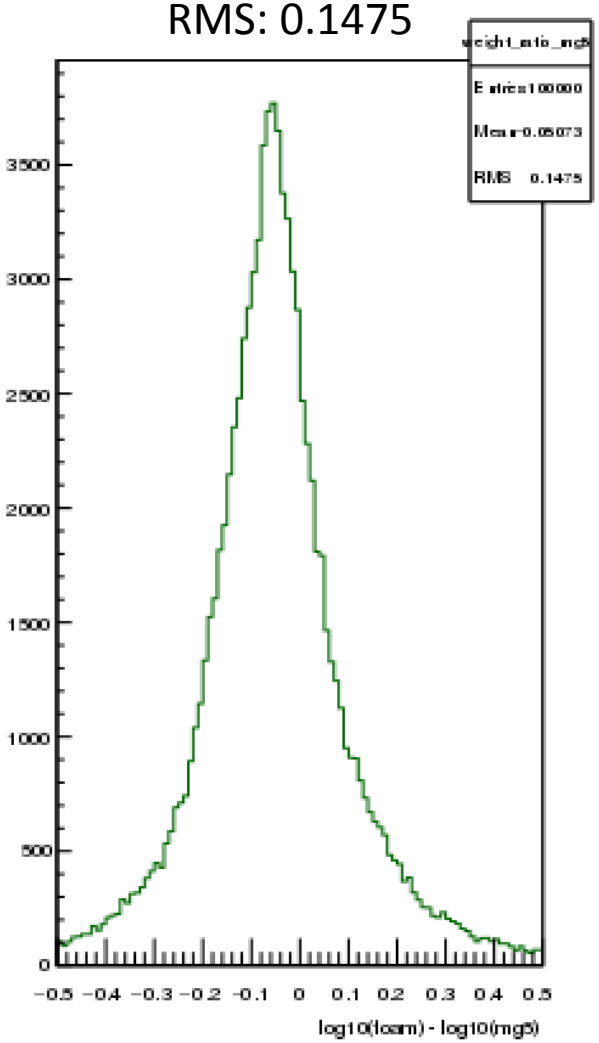


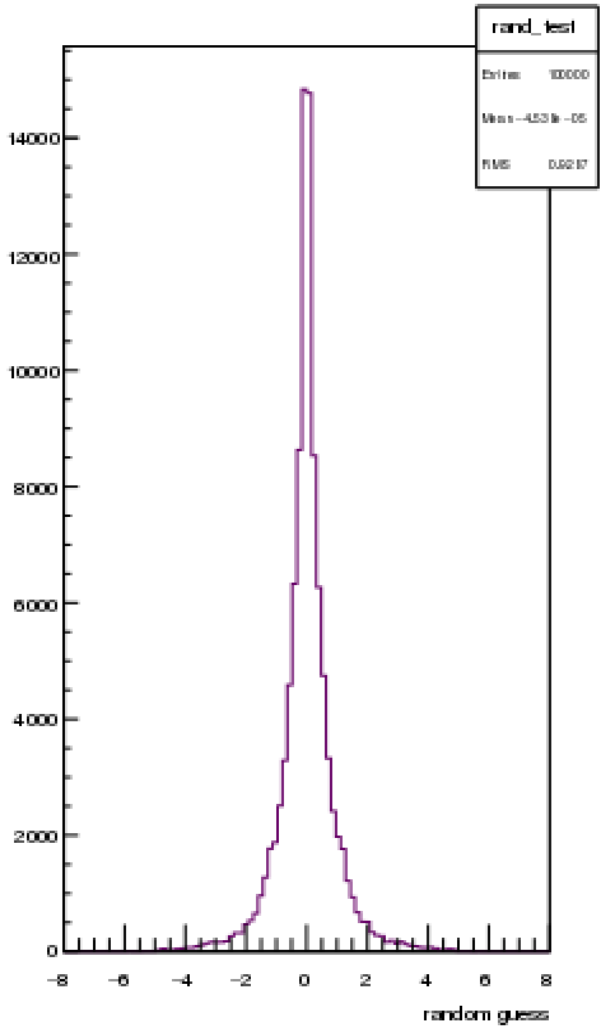


Foam Weight Comparison: Linear Kernel

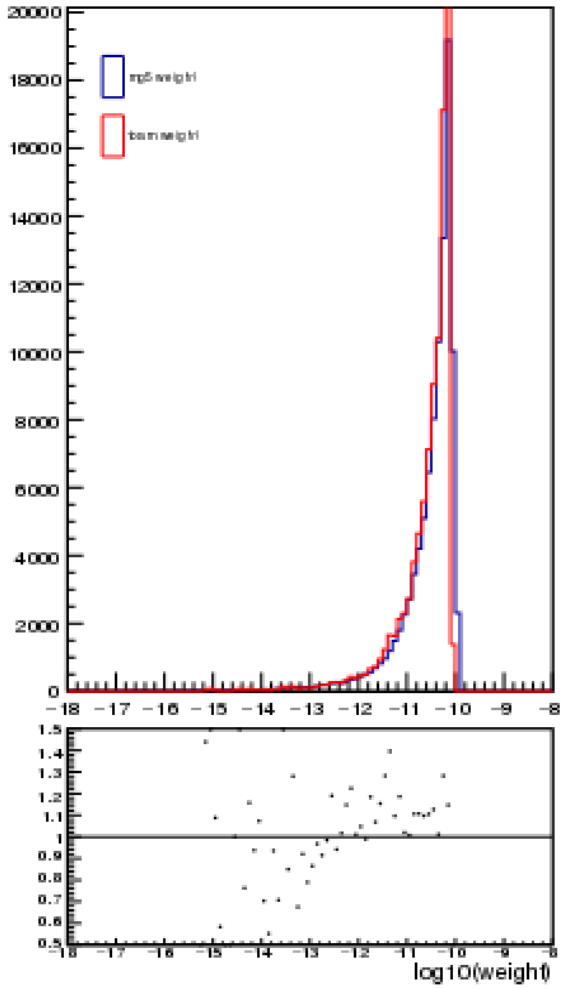
RMS: 0.1475



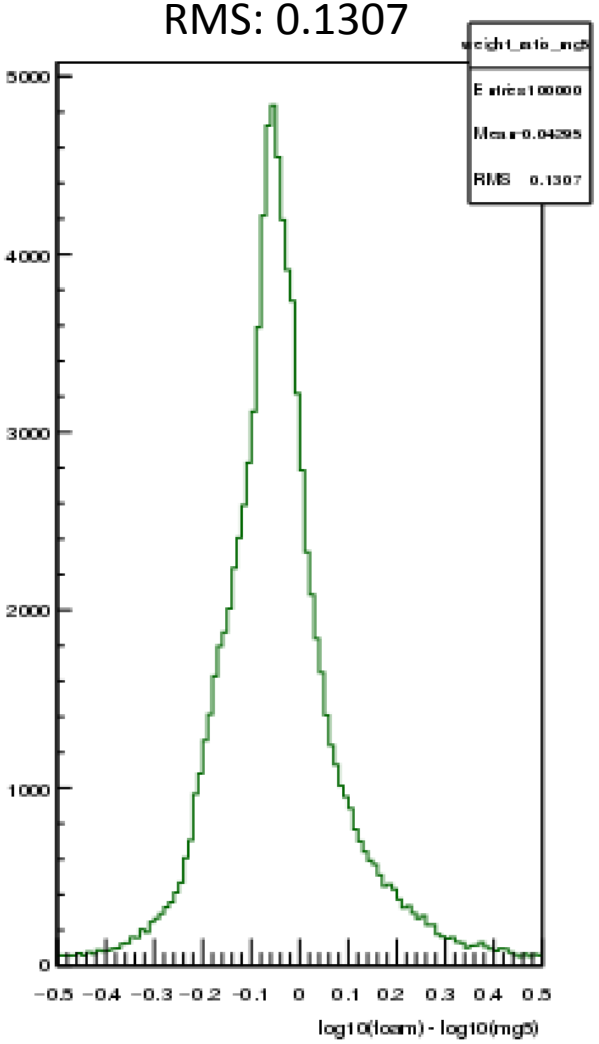


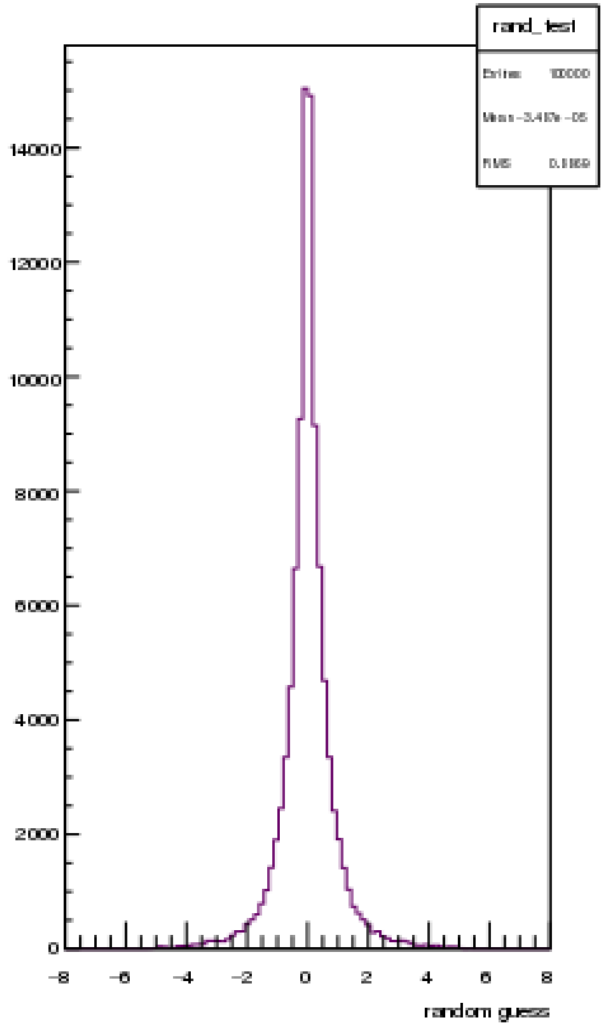


Foam Weight Comparison: Gaussian Kernal



RMS: 0.1307





Summary

- Each optimization of the foam attempts to improve the accuracy of the likelihood assigned to an event using the foam as a lookup table
- By creating multiple foams for ttH(bb) and ttbar, we can assign an event to one of these two decay processes by simply comparing the ratio of likelihoods given by the two foams
- This will speed up variable discrimination, and could potentially make it more accurate, as foams could be created with more precision than the current matrix element method

European Immersion!

Me and foam!

Citations

Search for the Standard Model Higgs boson produced in association with top quarks and decaying into a bb pair in pp collisions at sqrt(s) = 13 TeV with the ATLAS detector. The Atlas Collaboration, 2016

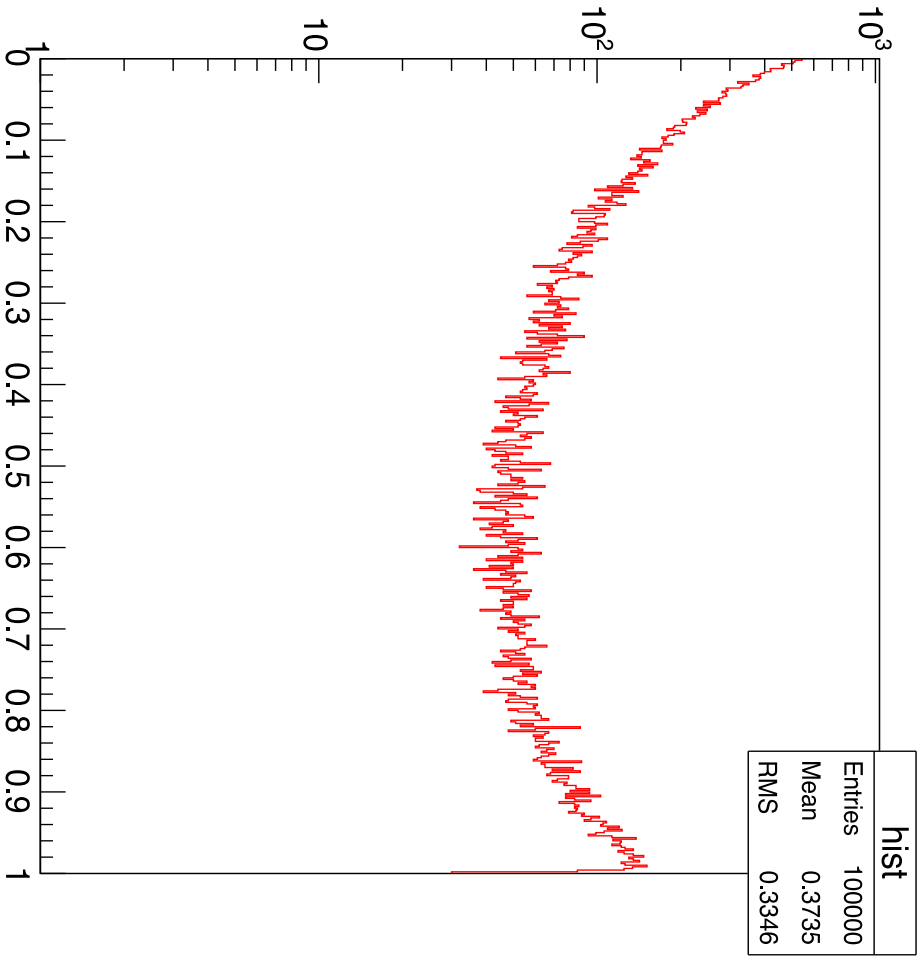
Foam: A General-Purpose Cellular Monte Carlo Event Generator. S. Jadach, 2002

PDE-Foam – a probability-density estimation method using self-adapting phase-space binning. Dannheim et al., 2008

MadWeight in C++. A. Mertens, S. Wertz, 2015

Backup Slides

Distribution of transformed variable in terms of foam coordinates, ttbar events



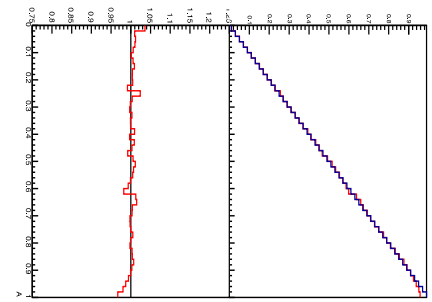
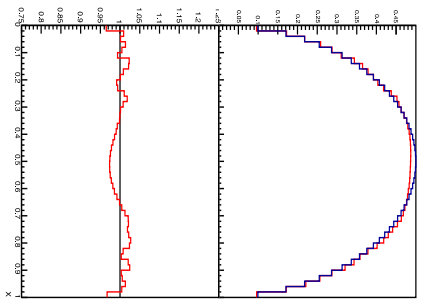
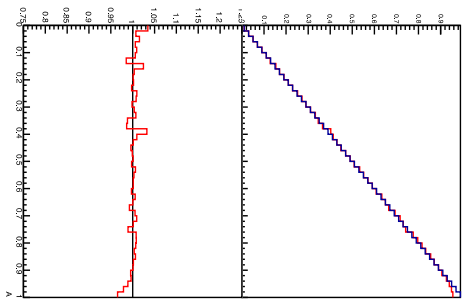
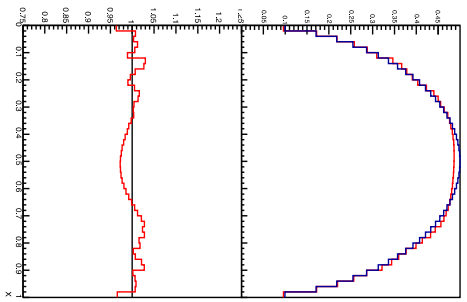
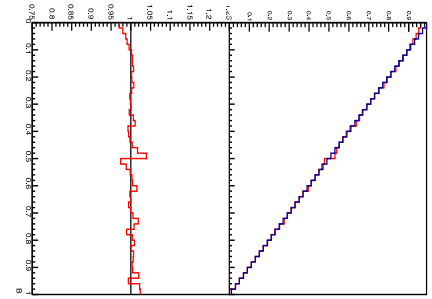
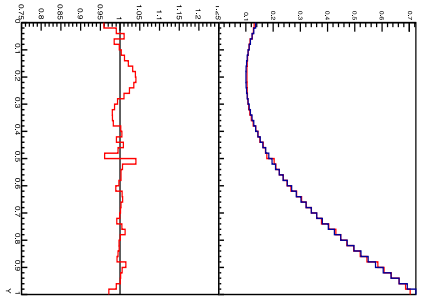
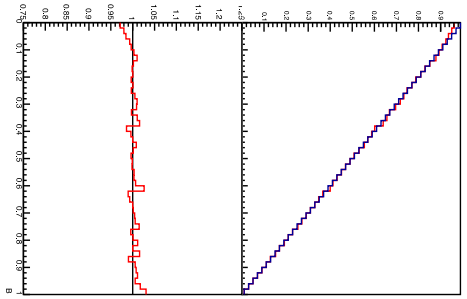
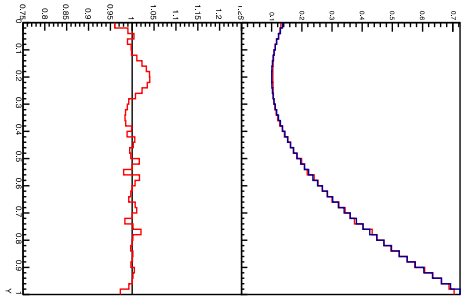
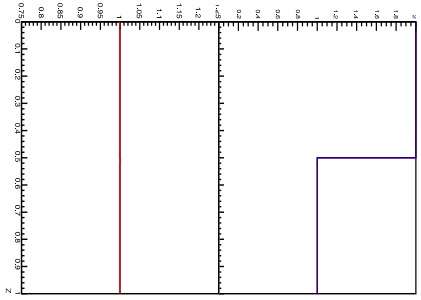
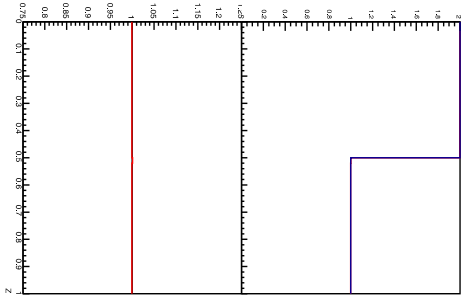
TFoam vs. PDEFoam

- There are two different implementations of foam, called TFoam and PDEFoam
- All of my work up to this point was with TFoam, which works with a “density function” to estimate the integral of each cell using Monte Carlo integration
- PDEFoam, instead, uses an event distribution, which each cell samples in order to define an integral

PDEFoam Kernel Implementation

- Only PDEFoam has the kernel built in function, which samples neighboring cells along with the one an event is found in, to generate a more accurate integral
- We want to use this kernel functionality while sampling from density functions instead of events, as TFoam does
- Thus, I added functionality for PDEFoam to sample from functions like TFoam

Tfoam vs. PDEFoam: 200,000 Cells

PDEFoam Optimization: Next Steps

- PDEFoam is much slower than TFoam right now when it works with events instead of functions
- However, with similar speeds, PDEFoam could provide even more accurate integrals than TFoam, which in turn would strengthen the accuracy of our classification of events as background or ttH(bb) events

Speeding up PDEFoam

- Right now, PDEFoam samples the integral of an cell by sampling Monte Carlo generated points, looking within a certain “box” around each event for events
- However, this is very slow, because each cell could sample the same events hundreds of time
- Instead, we want to sample the events within a cell only once, which would in theory give just as valuable of an integral in a fraction of the time
- I am working on this now, and hope to finish by Friday