# Toy Model Research Summary

Brandon McKinzie

Advisors: Peter Jacobs, Leonardo Milano

## I. Introduction

At the Large Hadron Collider (LHC), particles are accelerated in opposing directions at nearly the speed of light and collide into one another. Due to the enormous amounts of energy involved in such collisions, thousands of particles are able to be produced. Their properties are detected and analyzed in order to obtain knowledge about the underlying physics. For example, collisions of heavy nuclei are thought to produce a state of matter called the quark-gluon plasma (QGP) that existed just microseconds after the Big Bang. Since the plasma cannot be analyzed directly, we must study the properties of particles that came from this medium to infer about its properties. Another interesting byproduct of these collisions are very high-energy groups of particles, collectively called "jets," that are a consequence of the asymptotic freedom of QCD; quarks are not allowed to exist unbound (under normal conditions), and so when one is created in a high-energy collision, many more pairs of quarks are created. This process of fragmentation of the initial high-energy quark into a cluster of quark/anti-quark pairs and/or triplets results in a jet. Studying the properties of jets can provide clues about the nature of the QGP, and vice-versa. This relationship is the source of inspiration for my project.

I will begin by writing a "toy model" of particle collisions that includes the physics of interest and that results in the production of jets. This will be done exclusively in the C++ programming language with the aid of the ROOT data analysis framework created at CERN. This part of the analysis is based on the notion of 2-particle correlations, which says that for certain pairs of particles, their individual properties have a mutual relationship or connection. Here, this corresponds to selecting a high-energy particle (the "trigger" hadron) and scanning for the existence of a jet traveling in the opposite direction. By studying the properties of the resulting jet, depending on what I assign as input parameters, we can gain an understanding of the interactions involved throughout the collision process. There are, of course, a few assumptions that underlie this analysis, namely that (1) jets tend to be produced approximately back-to-back, and that (2) abnormally high energy particles are a good indicator of a jet. The challenge is reducing the overall signal/noise ratio, where a signal corresponds to correctly identifying a jet, and the

noise corresponds to clusters of particles that we erroneously call a jet.

The novelty of this project is in building a model from scratch that relies almost exclusively on sampling probability distributions from recorded data while also being capable of switching on/off physical process of interest, such as correlated hadron-jet pairs and the effects of elliptic flow. As of May 13, when this draft was written, there is still plenty of work to be done, and the project is far from over. This document thus serves as a summary of what has been done in the past few months with a brief discussion of the next steps to be taken.

# II. Theory

The model relies on a set of commonly used functional forms to describe distributions. Collision events are simulated via Monte Carlo sampling of measured and/or known probability distributions. The main particle properties we will be studying are transverse momentum $p_T$, $\eta$, and $\phi$. The pseudorapidity, denoted $\eta$, is roughly equivalent to rapidity when $p_T >> m$. It is convenient in that particle production is approximately uniform over the possible values of $\eta$, which are constrained by detector geometry, since

$$\eta \equiv -ln\Big[tan\Big(\frac{\theta}{2}\Big)\Big] \tag{1}$$

which shows the functional dependence of $\eta$ on the polar angle $\theta$. The ALICE detector provides good detection in $|\eta| < 1$, and we will restrict our analysis to charged particles at mid-rapidity, $|\eta| < 0.8$ to account for any undesirable fringe effects.

The most naïve approach would assume uniformity in both $\eta$ and $\varphi$. However, when two nuclei collide with a non-zero impact parameter, the spatial overlap in the transverse plane no longer looks spherical, but can take on an almond shape of the form seen in figure 1 below. This results in a non-uniform density

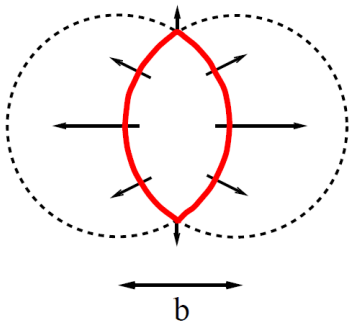



**Figure 1:** Schematic picture of elliptic flow in the transverse plane. Source: ref. [5].

distribution of particles, also called a spatial anisotropy. Soon thereafter, as the spatial anisotropy lessens, it is "converted" into a momentum anisotropy among the particles, which allows one to measure the degree of elliptic flow in each event. Specifically, the predicted number of particles produced at a given value of $\varphi$ is proportional to a quantity denoted as $v_2$, where

$$\frac{dN}{d\varphi} \propto 1 + 2v_2 cos[2(\varphi - \psi_{EP})] \tag{2}$$

and $\psi_{EP}$ represents the angle of the event-plane in $\varphi$.

Somewhat similarly, the analysis of two-particle correlations is done using distributions of the *differences* between two particles' properties. For example rather than plotting $dN/d\varphi$ of particles, we would plot $dN/d\Delta\varphi$ where $\Delta\varphi$ is between a specific trigger particle and all others. If any correlations exist, we would expect to observe a particular structure in such plots for certain regions of the phase space.

There is an observed suppression of hadron production in heavy-ion collisions as compared to independent superpositions of nucleon-nucleon collisions. This is expected to be due to energy loss of the partons as they propagate through the hot and dense QCD medium [1]. To quantify nuclear medium effects at high $p_T$, the so called nuclear modification factor $R_{AA}$ is used. Defined as ratio of charged particle yield in Pb-Pb to pp, scaled by number of binary nucleon-nucleon collisions:

$$R_{AA}(p_T) = \frac{(1/N_{evt}^{AA})d^2N_{ch}^{AA}/d\eta dp_T}{\langle N_{coll}\rangle (1/N_{evt}^{pp})d^2N_{ch}^{pp}/d\eta dp_T} \tag{3}$$

where we use the measured value of $\langle N_{coll}\rangle = 1690 \pm 131$ for 0 - 5% centrality.

## Jet Reconstruction

The two essential stages for any jet algorithm involve (1) identifying the objects belonging to a cluster, and (2) calculating the kinematic variables that define the jet from the objects defining the cluster [3]. The clustering algorithm used here, the anti-$k_T$ algorithm, relies on the association of measured particles based on their transverse momentum[1]

Before defining what is meant by the "anti" in anti-$k_T$, let's consider the (longitudinally-invariant) $k_t$ algorithm for hadron collider. The algorithm is initialized with a vector of partons from the event, and each is considered as a proto(or pseudo)-jet. Then, the following distance measures are computed for each parton/parton pair: (1) $d_{ij}$ between all pairs of particles $i$ and $j$, and (2) $d_{iB} = p_{ti}^2$ between every particle $i$

[1]This is contrast with clustering done by cone algorithms, which associate particles based on angles relative to a jet axis.

and the beam, where

$$d_{ij} = min(p_{ti}^2, p_{tj}^2)\frac{\Delta R_{ij}^2}{R^2} \tag{4}$$

and $\Delta R_{ij}^2 = (y_i - y_j)^2 + (\varphi_i - \varphi_j)^2$ is the distance in $(y, \varphi)$ space between $i$ and $j$. The parameter $R$, informally referred to as the "jet radius," determines the angular reach of the algorithm [7]. If the minimum value of these quantities is a $d_{iB}$, then $i$ is designated as a jet; if it is a $d_{ij}$, then $i$ and $j$ are combined into a single proto-jet by summing their four-vector components. If two particles/objects are merged, they are replaced in the list by the object resulting from summing their four-vectors [4]. This is repeated until all proto-jets have become jets. Mathematically, the algorithm terminates when the following evaluates as true: $(!\exists i, j)(k_{T,(i,j)}^2 < k_{T,i}^2)$.

The *anti-$k_t$* jet algorithm uses the distance measures

$$d_{ij} = min(1/p_{ti}^2, 1/p_{tj}^2)\frac{\Delta R_{ij}^2}{R^2} \tag{5a}$$

$$d_{iB} = 1/p_{ti}^2 \tag{5b}$$

, it behaves in some sense like a "perfect" cone algorithm , in that its ard jets are exactly circular on the $y - \phi$ cylinder. We can see that the consequence of inverting all the values (e.g. $1/p_T$) is that the highest-$p_T$ objects are clustered first.

One way of correcting for background contamination of hard jets involves the use of jet "areas," which provide a measure of a given jet's susceptibility to soft contamination. Jet areas can be determined by examining the clustering of a large number of additional infinitesimally soft "ghost" particles. Inclusive jets correspond to all objects that have undergone a "beam" clustering ($d_{iB}$ recombination).

# III. Particle Generation

Various implementations responsible for generating the particles were explored throughout the course of the semester. First, an overview will be given of our initial approach, and then a more detailed account will be given on the current methods employed for particle generation. The first steps were to sample some (as of yet undetermined) number of particles uniformly within $|\eta| < 1$ and $0 < \varphi < 2\pi$. Initially, the model used the following $p_T$ sampling distribution, courtesy of Redmer Alexander Bertens.

$$P(x) \propto A(B^2 x e^{-Bx}) + (x > 1)C\Big[0.12\sqrt{D^2 + x^2}\Big(1 + \frac{1}{(0.82)(E)}\sqrt{D^2 + x^2}\Big)^{-E}\Big]\Big(\frac{1}{1 + e^{-\frac{x-F}{G}}}\Big) \tag{6}$$

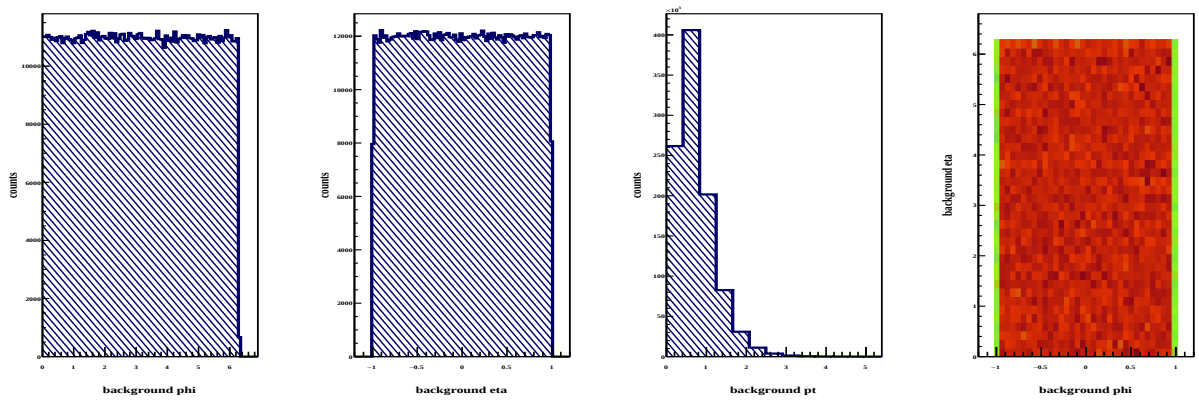



**Figure 2:** Basic distributions for single particles sampled in *phi*, $\eta$, $p_t$, and *phi* vs. *eta*.

with the fitting parameters, denoted by capital letters, initialized with the values in table 1. With the help of the ROOT libraries, this simple implementation alone was able to plot basic particle spectra as seen in figure **??**.

At this stage, such particles were referred to as "background" particles, as they were sampled randomly from a (generally) soft distribution. Since we are interested in two-particle correlations associated with jet production, we also designate a single high-$p_T$ particle per event as a "trigger" particle, with a corresponding "associated" particle opposite to it in $\eta$ and $\phi$. The associated particle's phi is also Gaussian smeared, centered at $\pi$ radians from the trigger with some standard deviation $\sigma \approx 0.2$. The trigger-associated particle pairs will later serve as what is known as an h+Jet pair. These assumptions are made from the perspective of two-particle correlations, which stem from the notion that, in the early stages of the collision, particles can be produced back-to-back with very high momenta.

Next, we incorporate a non-zero $v_2$ parameter, which varies as a function of the given particle's $p_T$, that provides information regarding elliptic flow. Specifically, it is known that, in the presence of flow, particles are not generated uniformly in $\phi$, but rather they exhibit a dependence on $v_2$ in the form,

$$\frac{dN}{d\phi} \propto 1 + 2v_2 cos[2(\phi - \psi_{EP})] \tag{7}$$

where $\psi_{EP}$ represents the angle of the event-plane in $\phi$. For now, we simplify this by setting $\psi_{EP} = 0$.

| A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|
| $2.43 \times 10^9$ | 2.99 | $1.01 \times 10^7$ | 0.14 | 5.6 | 2.8 | 0.2 |

**Table 1:** Initial values of fit parameters for the first sampling $p_T$ distribution used by the model.

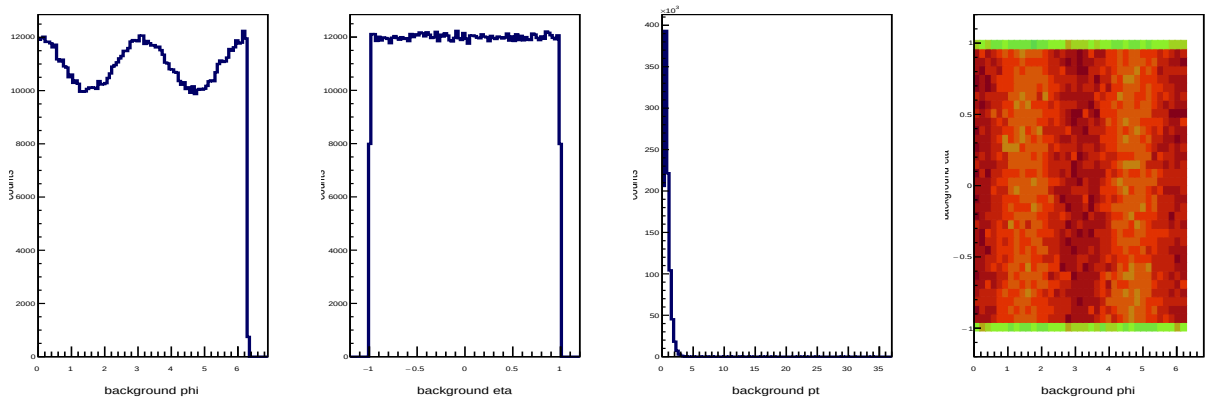



**Figure 3:** Trigger particle distributions for $\phi$, $\eta$, $p_T$, and $\eta$ vs. $\phi$.

The resulting change in the background distribution over *phi* can be seen in figure **??**. To be safe, we assume that $v_2$ can takes on a different value for trigger, associated, and background particles. Having the ability to alter the $v_2$ parameter for each category also provides us the opportunity to explore how the event outcomes depend on such a difference. In effect, we have introduced a new independent variable that we can tune as we please to observe how it affects the physics and/or distributions. A general goal of the model is to eventually have as many tunable parameters than be switched on/off as the user pleases.

Now that we have the basic phase space information regarding each particle category, we can begin plotting combinations of these variables. Of particular interest is the difference in phi of the trigger and associated particle. We expect this distribution to be centered and peaked at $\Delta\phi = \pi$ with a roughly Gaussian spread, since we have essentially defined it as such. For similar reasons, introduction of elliptic

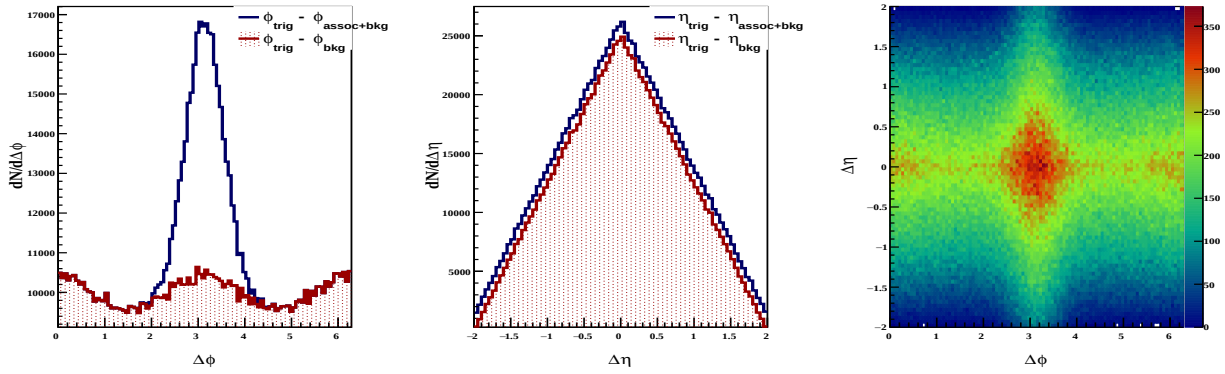



**Figure 4:** Plots of $\Delta\phi$ and $\Delta\eta$ between the trigger particle and all others. The results for background particles are plotted in red, and the differences for associated particles are plotted in blue. Note that these distributions, collected in the earliest stages of the project, correspond to events where similar amounts of trigger particles as background particles were generated, so as to easily examine their different properties.

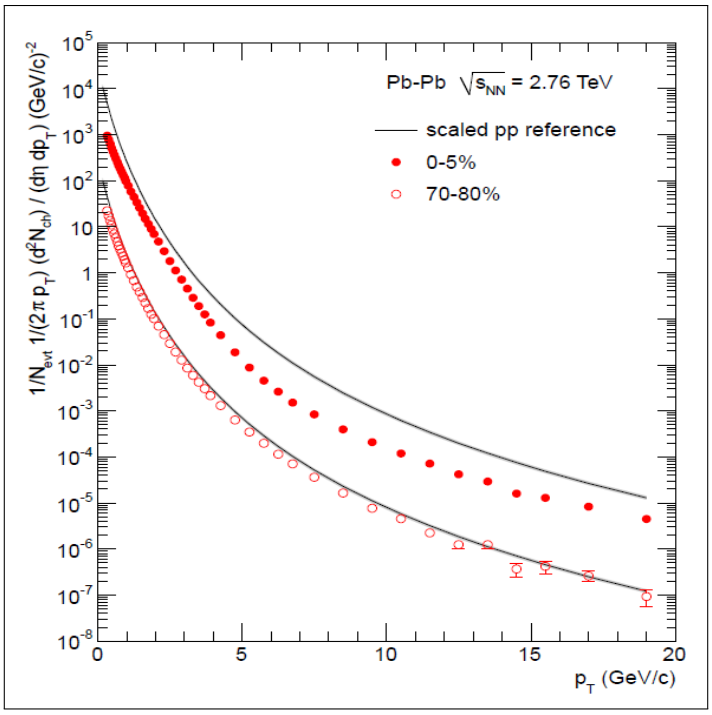



**Figure 5:** [Source: Figure 2 of ref. [1]]The pT distributions of primary charged particles at mid-rapidity ($|h| < 0.8$) in central (0–5%) and peripheral (70–80%) Pb-Pb collisions at $\sqrt{s_{NN}}$ = 2.76 TeV. Error bars are statistical only. The systematic data errors are smaller than the symbols. The scaled pp references are shown as the two curves, the upper for 0–5% centrality and the lower for 70–80%. The systematic uncertainties of the pp reference spectra are contained within the thickness of the line.

flow should not greatly alter this shape. However, if we also include differences between the trigger and background in this distribution, we expect the modulation introduced from a non-zero $v_2$ to show up as a modulation in the overall $\Delta\phi$ distribution. We show how v2 affects both (1) the differences between trigger and all particles and (2) the differences between just trigger and associated particles in figure 4. Since the trigger itself is more likely to be found in regions along the event-plane, one would expect it to be more probable that any given be particle be found at nearly the same $\varphi$ or directly opposite from it. The triangular shape of the $\delta\eta$ distribution is what one would expect for differences taken between pairs of variables all sampled uniformly over [-1, 1] (i.e. it is due to the geometry of our acceptance)

The current implementation relies almost exclusively on sampling from published data to generate particles and their properties. The $p_T$ distribution for the soft background, shown in figure **??**, is from the same ALICE 2010 data we will soon be comparing with [1]. This also provides us a way to check that our sampling behaves as expected, as we will be able to plot our model's $p_T$ distribution against the raw data after we apply the appropriate corrections and normalization factors. We now fit the published $p_T$ distribution with a simpler functional form proposed by Hagedorn to describe experimental hadron

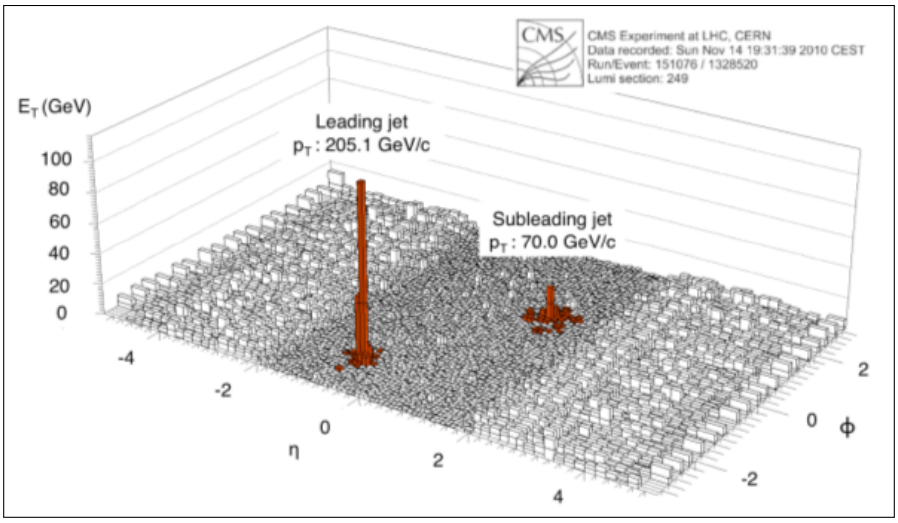



**Figure 6:** A CMS event containing a leading and subleading jet, indicated in red. Credit: Alexander Schmah.

production data as a function of $p_T$ over a wide range [6].

$$E\frac{d^3\sigma}{d^3p} = C\left(1 + \frac{p_T}{p_0}\right)^{-n} \tag{8}$$

Although simple in appearance, this function proved to fit the distributions quite well.[2]

# IV. Jet Finder

Now that we can properly generate basic events with high-multiplicity and particles exhibiting elliptic flow, we can begin searching for the existence of jets. The easiest way to begin this task is to artificially embed a jet in a known location in the eta-phi phase space and tell our jet finder to search for it; if it reports finding our embedded jet, and our embedded jet only, we have succeeded in writing an elementary jet finder. From there, we will improve it so that it can handle more realistic (and much more challenging) events.

After embedding a jet in our simulation over the background, we should expect to see something like Figure 6 upon plotting eta vs. phi. Any competent jet finder ought to be able to select the particles displayed in red, which can be distinguished by their abnormally high transverse energy. Although finding such outliers may seem like a straightforward task, we are seldom presented with the clean-cut distinctions seen in figure 6. More often, we will be tasked with determining which clusters of particles in a sea of possibly thousands of background tracks were produced by a hard scattering process.

[2]One unintended but amusing result was discovered upon plotting the ratio of our model over the reconstructed AOD data that deviated significantly from unity. It turned out that our model was sampling from the scaled pp reference in figure **??** and thus our ratio was actually $R_{AA}(p_T)$; we had verified the published dependence of $R_{AA}$ with $p_T$ on accident.

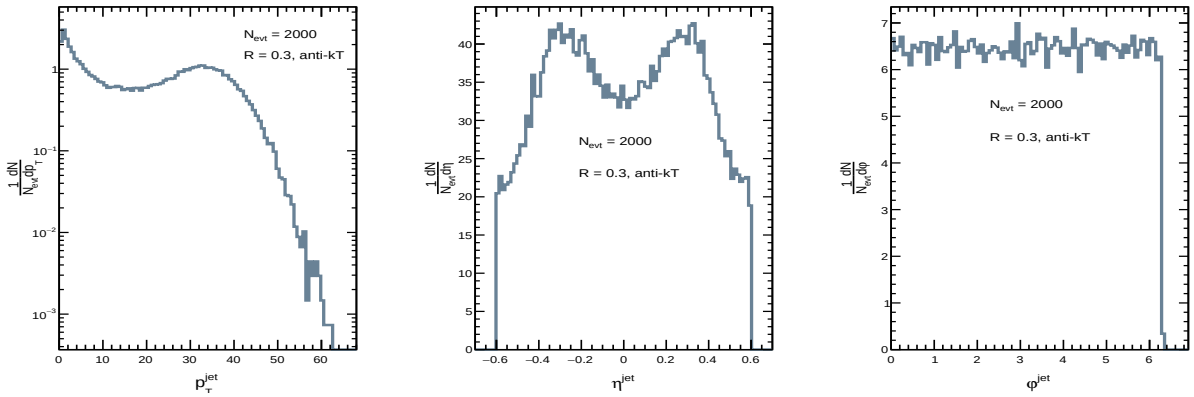



**Figure 7:** Basic jet quantities corresponding to the simplest simulation settings. Specifically, these correspond to 2000 generated events, each with (fixed) track multiplicity of 1667[3]. Tracks have sampled distributions like those seen in figure 2. Track acceptance is restricted to $|\eta| < 0.9$. No minimum value of jet $p_T$, typically denoted as $p_{T,min}$, is passed to FastJet.

The toy model both generates the events as well as analyzing them with the help of the FastJet framework. The particles are clustered using the anti-$k_T$ algorithm with a default value of $R = 0.3$ unless stated otherwise (we will soon explore how varying R can alter results). The clustering returns a vector of `PseudoJet` objects, which represent the proto-jets discussed earlier in the Theory section. To begin we explore various scenarios/checks on the jet finder to ensure quality performance. For now, we analyze the case of model particle generation with (a) $v_2$ set to 0, (b) track multiplicity constant for all events, and (c) a 100% tracking efficiency (i.e. all simulated particles are successfully "detected"). This is important for developing a baseline reference before incorporating the more complex model parameters.

The simplest results are shown in figure 7 with all relevant details regarding how the events were generated addressed in the caption. It is important to note that, since no transverse momentum cut is applied to the accepted jets, the number of jets found in the events are not infrared safe quantities [7]. For now, this cut (which is typically included) is omitted because it will be useful for estimating the soft background/underlying event.

Although the $p_T$ and $\varphi$ distributions match expectations[4], $\eta$ will prove to exhibit unanticipated behavior throughout the remainder of this report. For this reason, we will scrutinize it in greater detail as an attempt to discover the root cause of its shape. In theory, we would expect the jet $\eta$ distribution to be flat, as is the case for the single-particle tracks; there is no reason *a priori* that jets should be detected preferentially in certain regions of $\eta$ but not others. The most likely explanation, and the explanation that I will make a case for throughout the report, is that the strange shape is ultimately a geometrical consequence of clustering

[3]This value is the measured charged particle multiplicity for collision centralities of 2.5% from ALICE data in reference [1].
[4]Given that no cuts have been applied at this stage.

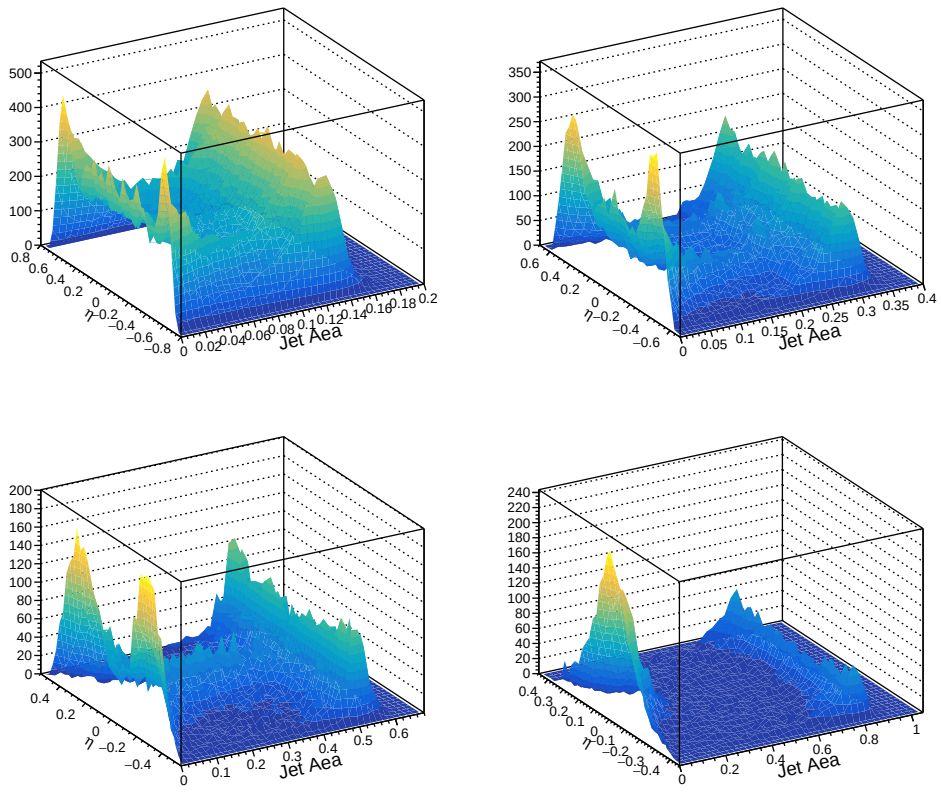



**Figure 8:** Surface plots of jet $\eta$ and $A$ for (top-left) $R = 0.2$, (top-right) $R = 0.3$, (lower-left) $R = 0.4$, and (lower-right) $R = 0.5$.

with the anti-$k_T$ algorithm restricted to a region of $\eta$ comparable to $R$, the angular reach of the algorithm.

First, we can see by inspecting the plots in figure 8 that distributions in $\eta$ deviate from expectations primarily for jets with areas close to zero. This is a direct consequence of how the clustering is performed by the anti-$k_T$ algorithm. Upon reviewing the equations in 5, we see that the clustering will evolve by sequentially grouping nearby high-p$_T$ tracks until their associated proto-jet has a large enough $p_T$ to be designated as a jet. Near the end of the clustering process, however, there will inevitably be small-area proto-jets scattered in ($\eta$, $\varphi$) that either don't meet the criteria to merge with a neighboring proto-jet or are situated such that all tracks within $R$ have already been assigned to a jet. Therefore, it is unsurprising to observe uniformity in $\eta$ for the large-area jets and less conventional behavior for small-are jets. There does appear to be a trend in the $\eta$ distribution for the low-area jets as $R$ increases that is a consequence of the boundary condition $|\eta_{trk}| < 0.9$. If our acceptance was larger, say $|\eta| < 2.0$, such boundary effects would be less noticeable, as can be seen in figure 9.

In figure 11a, distributions corresponding to the number of reconstructed jets per event are shown for different values of the input jet radius, $R$. Again, the number of jets depends on the clustering process

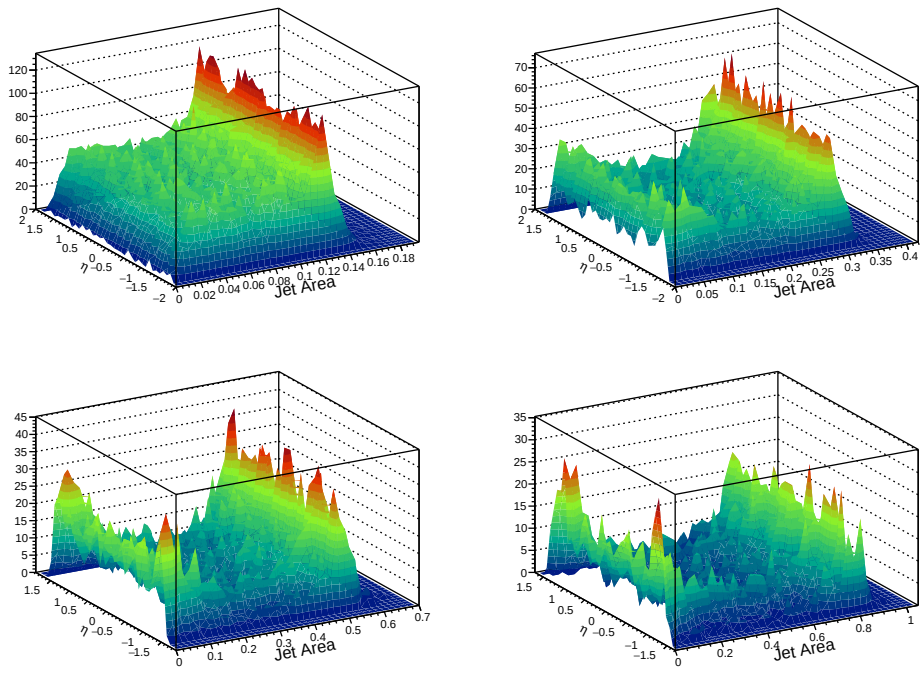
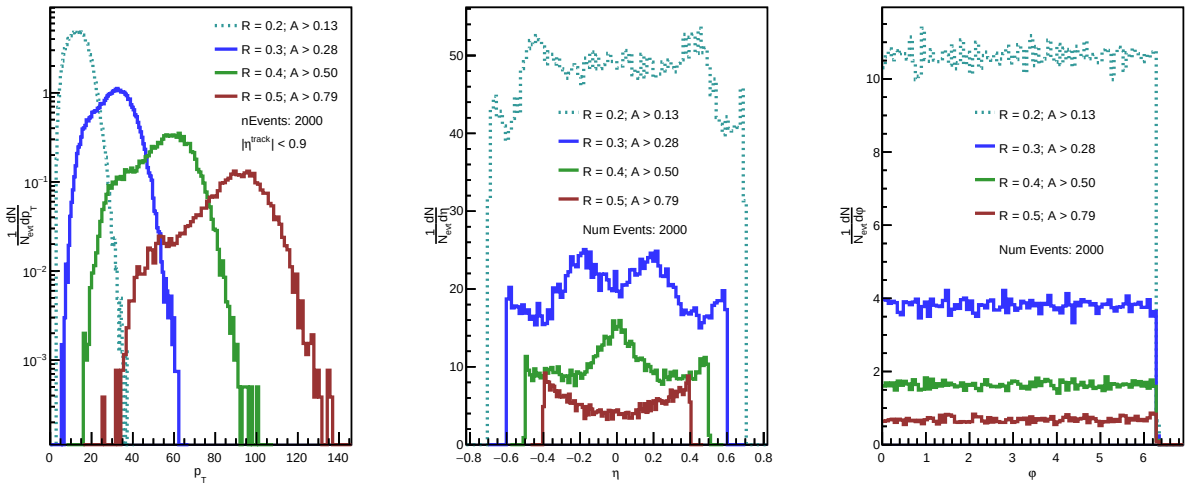



**Figure 9:** Jet area and $\eta$ distributions with an increased acceptance of $|\eta| < 2.0$. All plots correspond to model simulations of 200 events with (top-left) $R = 0.2$, (top-right) $R = 0.3$, (lower-left) $R = 0.4$, and (lower-right) $R = 0.5$.

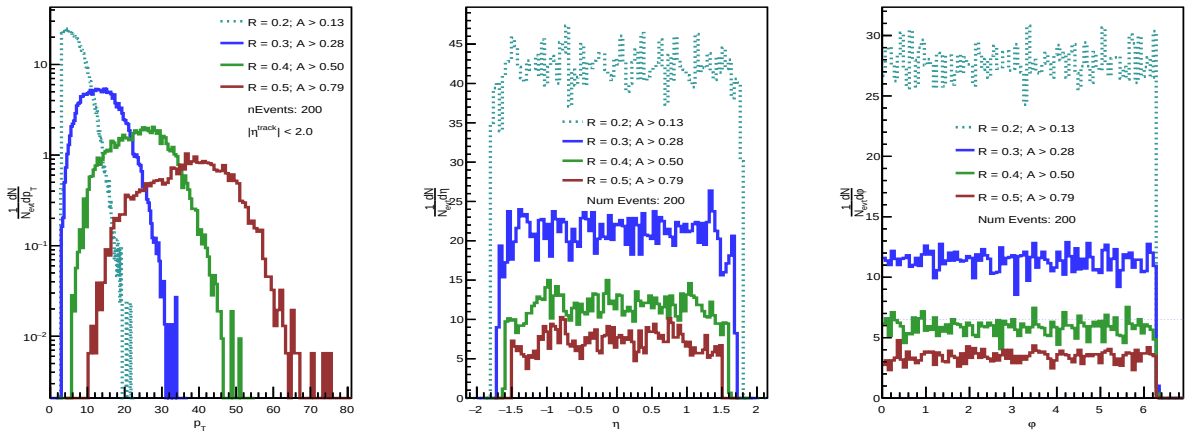
performed by the anti-$k_T$ algorithm. The distributions take the form of univariate Gaussians, as opposed to delta functions, due to the usual variability exhibited in Monte-Carlo sampling methods.

As the value of R increases, the number of measured jets per event decreases. This is nothing more than a consequence of geometry, since R determines the angular reach of the anti-k$_T$ algorithm in $\eta - \varphi$ space. Therefore, since no two reconstructed jets can overlap in $\eta - \varphi$, we have that large R corresponds to less reconstructed jets per event. These geometrical/boundary-condition effects will be evident in many of the plots that follow, such as in figure 11b.

Henceforth, a cut is applied required $A > a(R)$, where the value of the minimum threshold, $a(R)$, depends on the value of $R$. They are determined, for a given value of R, by visually inspecting the jet area distribution for the local minimum between the two outer peaks. This cut serves to further reduce background, as can be seen in the right-hand side plot of figure 12. It is clear that rejecting the jets with low area results in a lower proportion of jet candidates with $p_T = \rho A$. This can be understood when considering that lower-area jets are reconstructed near the end of the clustering algorithm, when the highest-p$_T$ tracks have already been clustered and the number of tracks within the algorithm's angular reach has decreased. Since we are seeking real jets resulting from hard processes, we reject the low-area jets due to their high probability of consisting of background particles. The resulting background-subtracted $p_T$ distributions for




**(a)** (Left to right) Distributions for jet $p_T$, $\eta$, and $\varphi$.

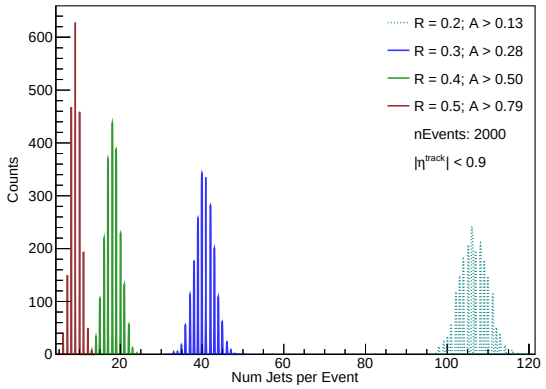



**(b)** (Left to right) Distributions for jet $p_T$, $\eta$, and $\varphi$.

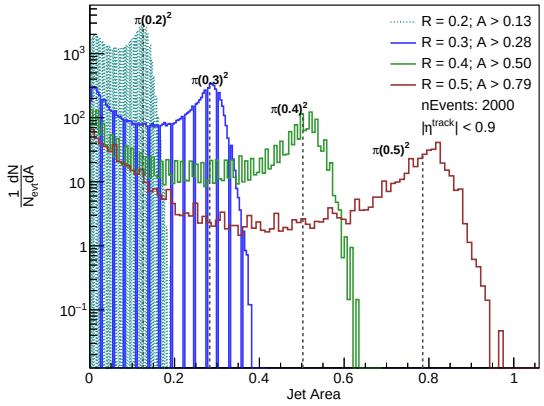
**Figure 10:** 10a shows distributions, each for various values of $R$, using the same parameters $|\eta| < 0.9$ and $N_{evt} = 2000$. 10b shows the result of increasing to $|\eta| < 2.0$.

different values of R are shown in figure 13.

As R increases, the width of distribution increases, meaning an increased proportion of jets with $p_T$ significantly different than the estimated background. Notice that as R approaches zero, the distributions should tend toward those of the single-particle tracks.

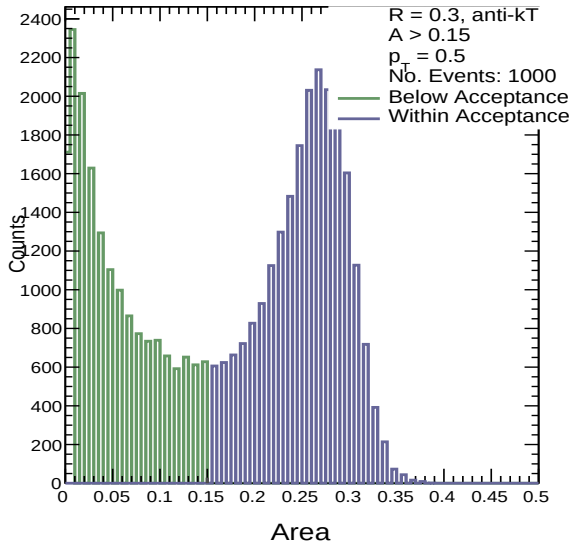



**(a)** The number of jets per event reconstructed by FastJet, shown for different input values of R.




**(b)** The event-normalized distributions of reconstructed jet areas for different values of R. The dashed vertical lines indicate where Jet Area = $\pi R^2$ for each value of R.

**Figure 11:** Illustration of how jet quantities can vary with R. (a) Number of reconstructed jets per event. (b) Distribution of reconstructed jet areas.



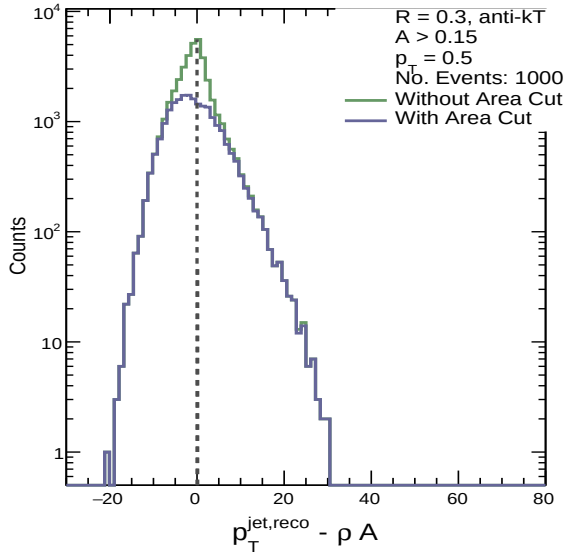




**Figure 12:** Illustration of how the jet area cut can affect $p_T$ distribution for R = 0.3. Left: An area cut requiring A > 0.15 is applied. Blue represents the region that is kept, while green represents the region rejected. Right: background-subtracted jet $p_T$ distribution shown for jets that passed that area cut (blue), as well as all jets where no area cut was applied (green).

# V. Incorporating Data

We use raw ALICE AOD data from (insert year here) that needs to be corrected for non-perfect detector efficiency $\varepsilon$. The specific efficiency we are concerned with here is $\varepsilon = \frac{N_{reco}}{N_{gen}}$, which is the fraction of particles

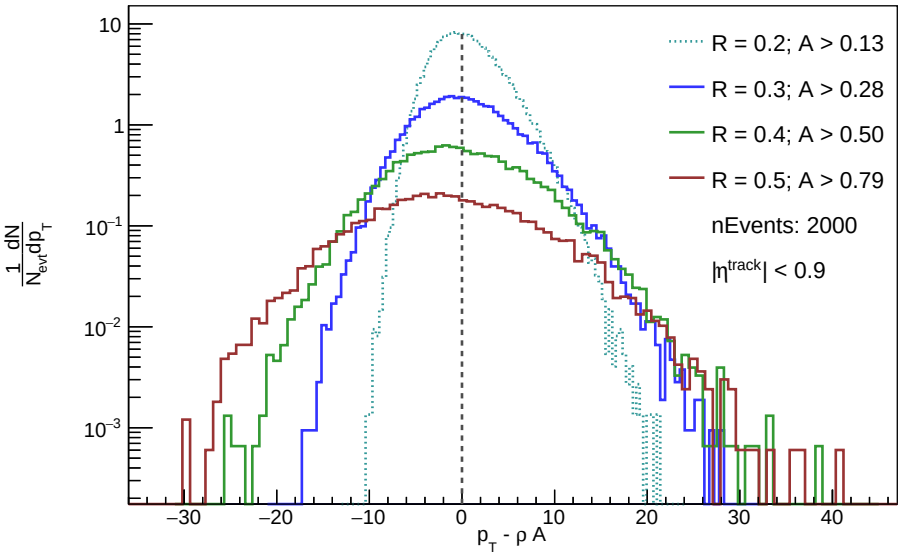



**Figure 13:** Illustration of how the jet area cut can affect $p_T$ distribution for R = 0.3. Left: An area cut requiring A > 0.15 is applied. Blue represents the region that is kept, while green represents the region rejected. Right: background-subtracted jet $p_T$ distribution shown for jets that passed that area cut (blue), as well as all jets where no area cut was applied (green).

that were reconstructed (i.e. found in the data) out of the number actually generated in the collision. This is required because not all particles created during the collision are perfectly reconstructed/detected at all. Since we are aware of this shortcoming, we apply $1/\varepsilon$ to the data so as to obtain the true distributions/numbers of particles corresponding to physical reality. Luckily, this process is so common that results from Monte Carlo simulations for specific datasets are readily available that provide the user with efficiency values as a function of particle variables. In particular, we apply this $1\varepsilon$ weight to each track, given the value of the track's $p_t$ and $\eta$ as well as the event centrality and the primary collision z-vertex. Another approach, as seen in figure 14 is leaving the AOD data the same (after applying normalization factors) and instead applying $\varepsilon$ to the *model*. What this means conceptually is comparing the data, which contains missing tracks lost due to the detector efficiency, with a version of the model wherein generated particles have some probability $\varepsilon$ of not being recorded. Note that $\varepsilon$ here is still a function of the particle properties in phase space, so this probability changes as the model considers new particles. The main takeaway from figure 14 is the excellent agreement between the published ALICE data and the reconstructed AOD data. The model (blue) sits above the rest due to normalization issues. Specifically, the model could only sample particle multiplicities starting at 2.5% centrality, since that was the lowest data point available.

Since this discrepancy was a valid physical result, new methods were needed to align the model properly with observed data. One approach was only comparing with data points above $p_T = 0.5$ and modifying

the model to generate events with charged-particle multiplicities given by

$$\frac{(1.8)(2.0*\pi)}{N_{evt}} \int_{0.5}^{20.} \frac{1}{2.\pi p_T} \frac{d^2N}{d\eta dp_T} \mathrm{d}p_T \tag{9}$$

However, one sees that there is a factor of $\frac{1}{p_T}$ "embedded" within the data we sample from, and it doesn't appear that there is any clean way of removing this $p_T$ dependent factor. The reason it is important to get the charged multiplicity value correct, on average, over events is because it directly impacts both (1) the histogram magnitudes after normalizations and (more importantly) (2) it effects the simulated efficiencies given by $\varepsilon$. Although it was rather trivial to apply $1/\varepsilon$ to the AOD data, the reverse method of applying $\varepsilon$

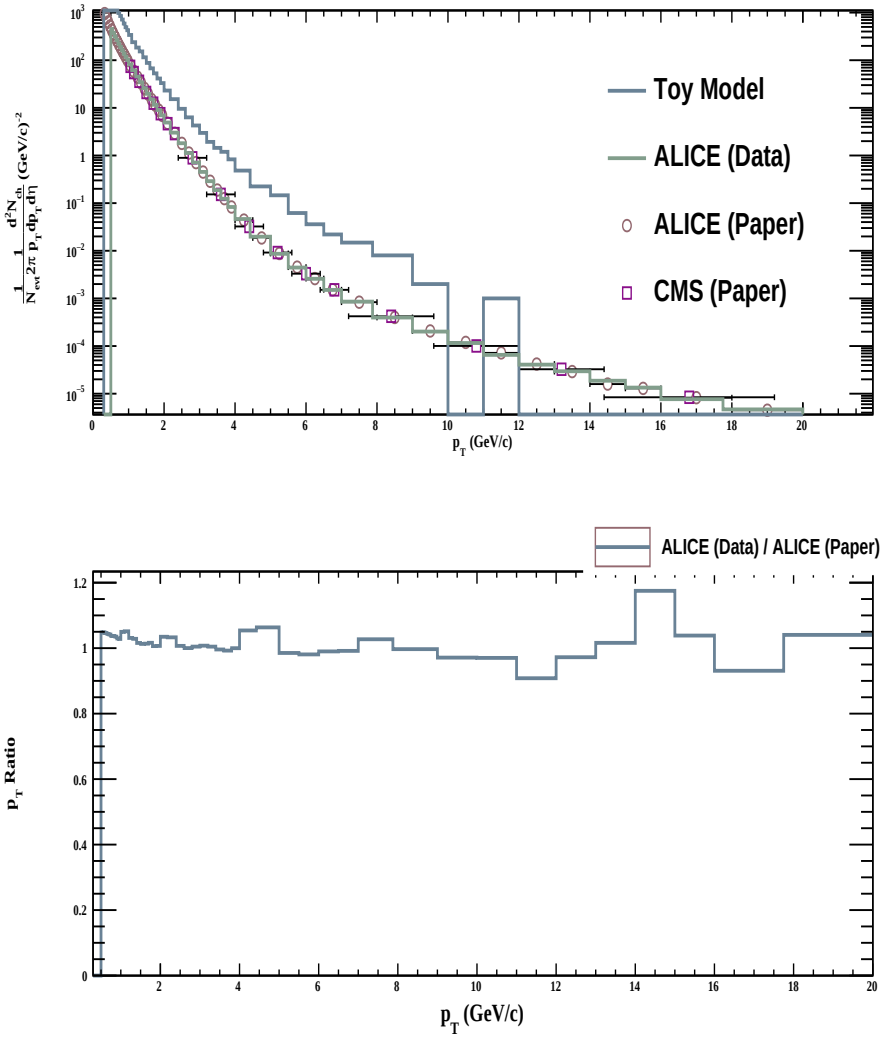



**Figure 14:** Comparisons between model and various forms of data. Now, we apply the *inefficiency* to the model instead of $1/\varepsilon$ to the raw AOD data.

to the model remains a problem.

# VI. What's Next

There are three major steps needed to complete this stage of the project.

1. Successfully simulate events with multiplicities given by $N_{gen} \cdot \varepsilon$ where $N_{gen}$ represents the multiplicity sampled as a function of centrality for fully corrected/published data. If the model can reproduce the same distributions as data, including this new addition of accounting for detector inefficiencies, it can be used to generalize for other detectors (given their efficiencies).

2. Implement h+Jet events in Pythia and merge them into the current model of the soft combinatorial background. The fastjet implementation is already in place and ready; all that is needed is a more complicated technique for introducing the jets themselves.

3. Subtracting out the background in the $p_T$ distributions by an amount $\delta p_T = p_T^{jet,reco} - p_T^{embed} - \rho A$, where $\rho$ is the background momentum density in $\eta - \phi$ space and A is the area of a jet cone given by the inputs to fastjet.

# VII. Conclusion

Although the project is far from over, this draft has explored some of our approaches for building a reliable toy model that is capable of generalizing well to new sets of data. Decisions are still being made with regards to the details of implementation, but the vast majority of the code base is in place and easy to modify. At present, the model consists of an EventGenerator to determine what high-level physical processes will occur (e.g. triggering and $v_2$ switches), a JetFinder designed to interface well with the EventGenerator and cluster on likely regions with jets, and many helpers that serve to make the jobs of EventGenerator and JetFinder as simple/abstracted as possible (e.g. an EventFunctions helper that takes care of the details of sampling/fitting/etc so the EventGenerator can focus solely on high-level event decisions). The next steps, as outlined in the previous section, are clear and in the process of being completed. Overall, the experience of building a model of ultrarelativistic collisions from scratch to where it is now has been rewarding and educational. I am incredibly grateful to Leonardo Milano, Peter Jacobs, and many other members of the ALICE collaboration for guiding me throughout this process and teaching me the seemingly countless subtleties involved in an experimental analysis.

# References


[1] ALICE Collaboration. Suppression of Charged Particle Production at Large Transverse Momentum in Central Pb-Pb Collisions at $\sqrt{s_{NN}}$ = 2.76 TeV. arXiv:1012.1004v1 *(nucl-ex)*, 2010.

[2] Alexander Schmah. Jet Production in Au+Au Collisions at STAR. Presentation.

[3] J. M. Campbell, J. W. Huston and W. J. Stirling, Rept. Prog. Phys. **70**, 89 (2007). [arXiv:hep-ph/0611148]

[4] S. Ellis, et al. Jets in Hadron-Hadron Collisions. (2007). arXiv:0712.2447.

[5] Ullrich Heinz. *Concepts in Heavy-Ion Physics*. Department of Physics, The Ohio State University, Columbus, OH 43210, USA.

[6] C. Wong and G. Wilk. Tsallis Fits to $p_T$ Spectra and Multiple Hard Scattering in pp Collisions at LHC. arXiv:1305.2627

[7] M. Cacciari et al., "Fastjet user manual"

# VIII. Appendix

## A. Emphasizing Quality Code

Overall, simplicity and ease of use was a primary goal in developing the codebase for this model. The model obeys the object-oriented paradigm and strives to use classes and methods with intuitive names and behavior. For example, by using the libraries that I've written, the following code is all that is needed to generate a single event and run the jet finder on the produced particles. Managing complexity and writing modular code is especially crucial when designing a model that may require new components to be built in some unknown order in the future.

```
eventGenerator->SetCentrality(2.5); // Cent = 2.5% is lowest available data point.
Printer::print("\tNumber of particles generated: ", eventGenerator->GetMultiplicity());

// Generate specified number/types of particles.
eventGenerator->Generate("bkg", (Int_t) eventGenerator->GetMultiplicity());
Printer::print("\tNumber of reconstructed particles: ", eventGenerator->GetRecoMult());

// Use ClusterSequence to get store list of jets in this event.
jetFinder->FindJets(eventGenerator->GetLastEvent());
Printer::print("\tNumber of jets found: ", jetFinder->GetNumJets());
```

**Figure 15:** Screenshot taken of the innermost loop of the toy model interface. The user can both control event selection and query the objects for desired event information. A static Printer class, tailored to accept various outputs of the model classes, is available for quickly examining outputs and debugging.

## B. More Figures

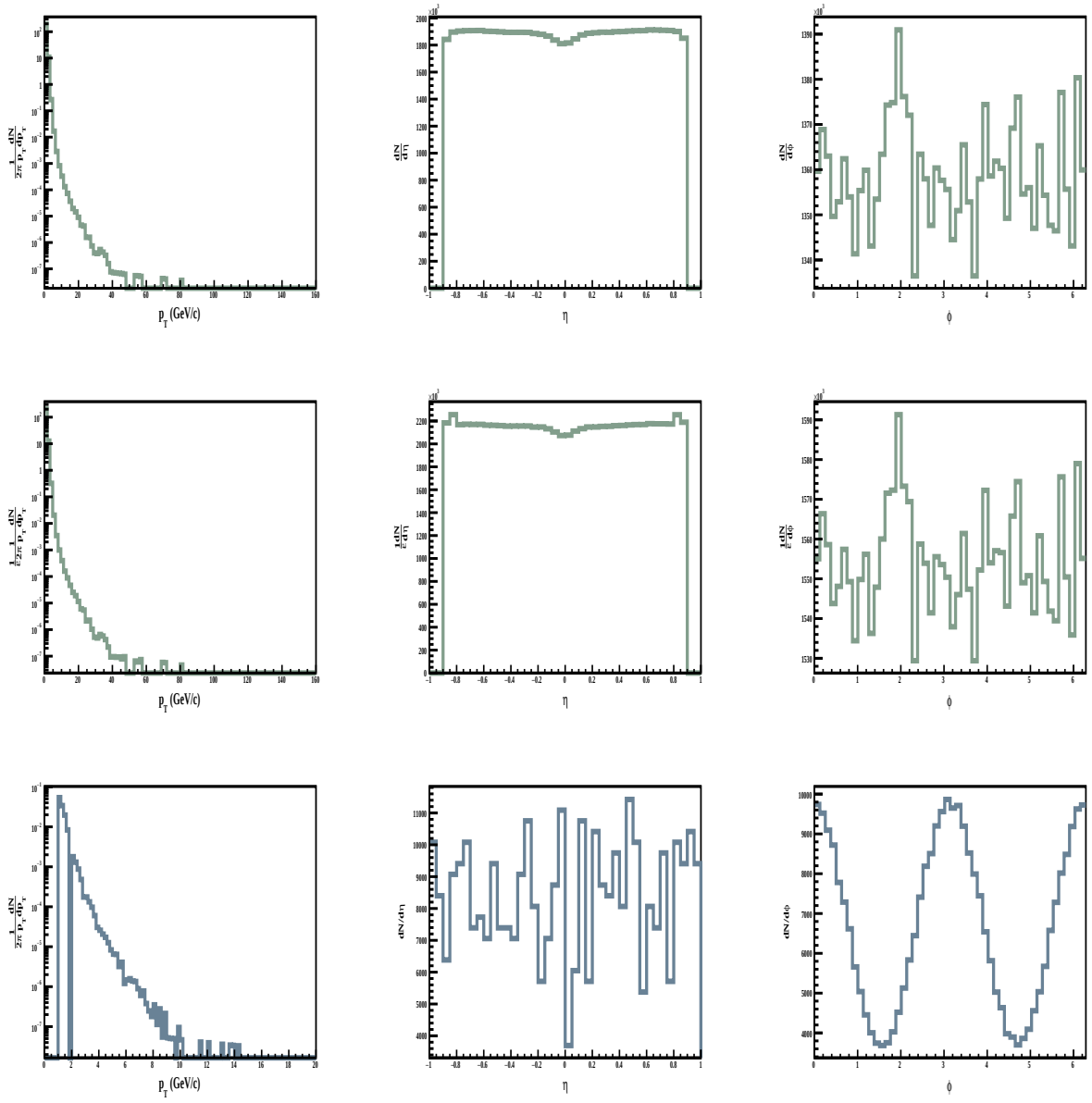



**Figure 16:** Comparison of $p_T$ distributions from reconstructed ALICE data, published ALICE data, published CMS data, and the toy model (which samples from the published ALICE data). Discrepancies are due to how multiplicity was obtained at this stage (almost fixed now) since only had access to 2.5% centrality and above. Also, min $p_T$ cutoff may play a role.

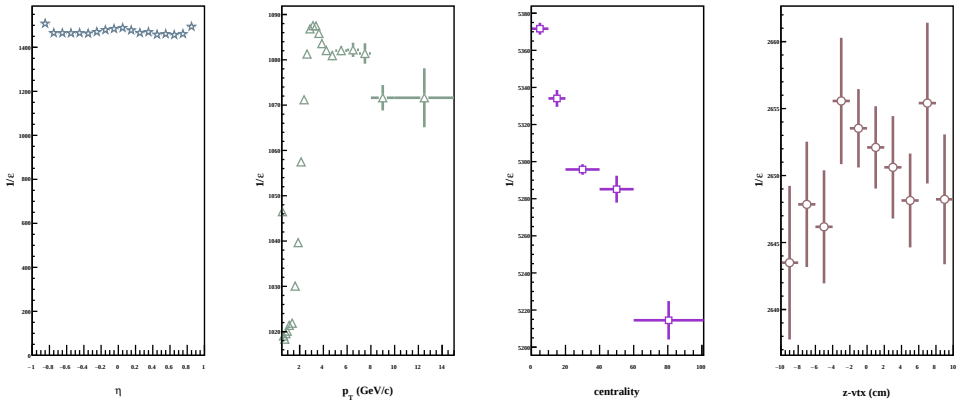



**Figure 17:** Edit me.

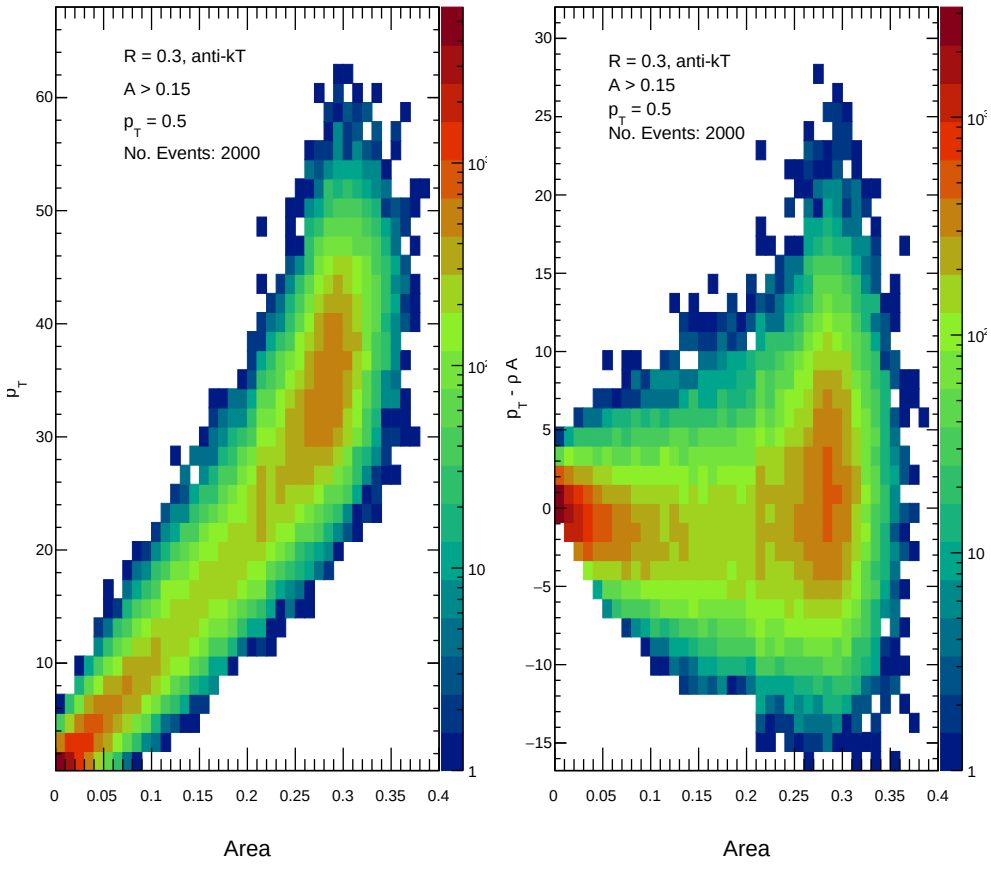
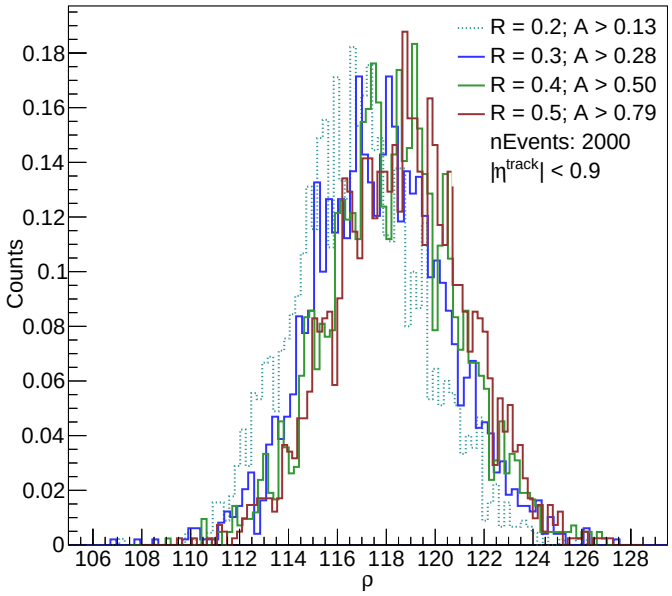
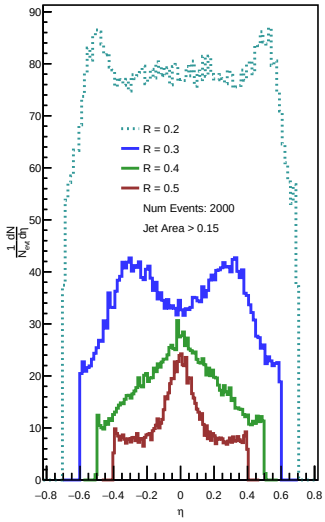
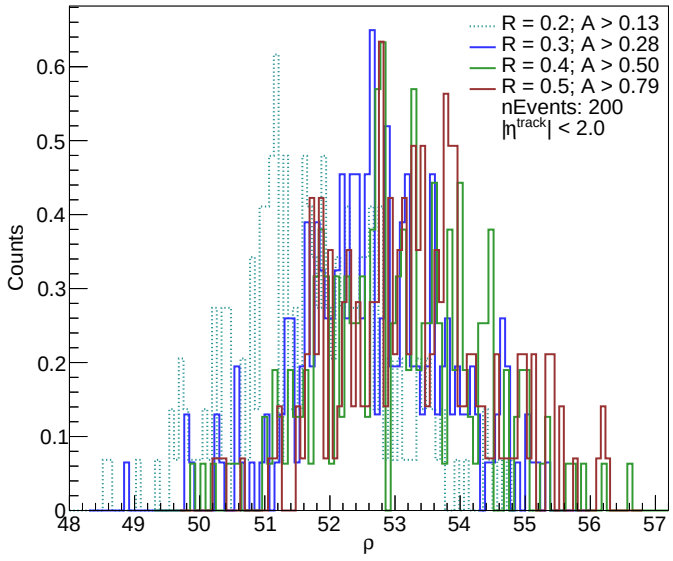
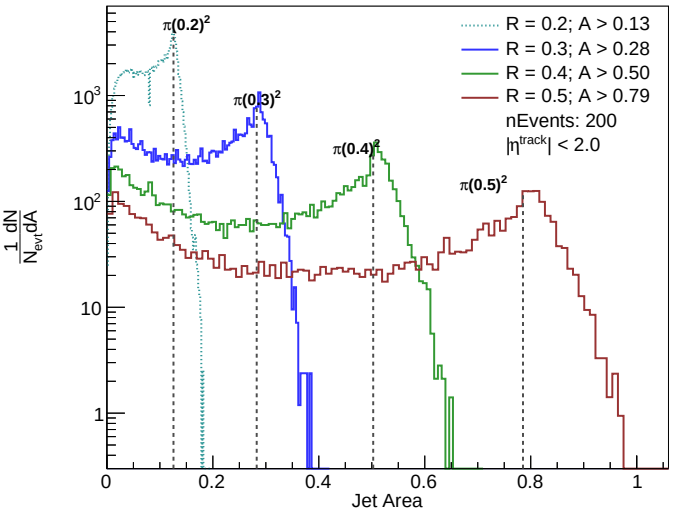


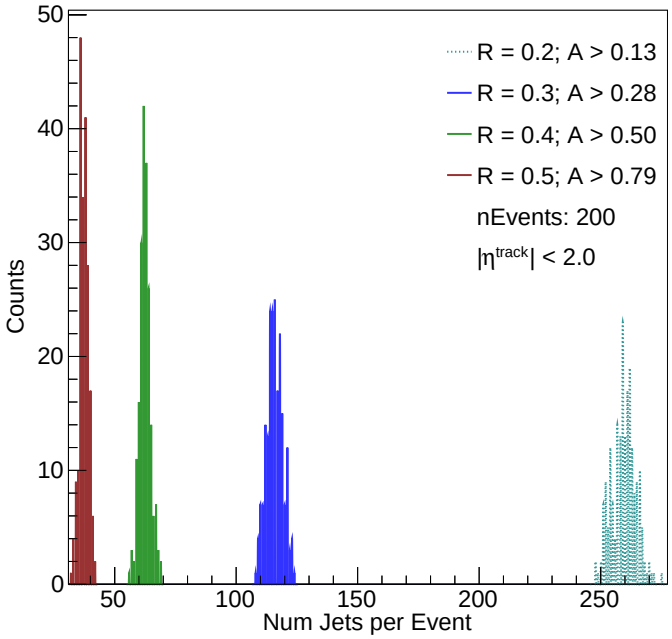

**Figure 18:** Edit me

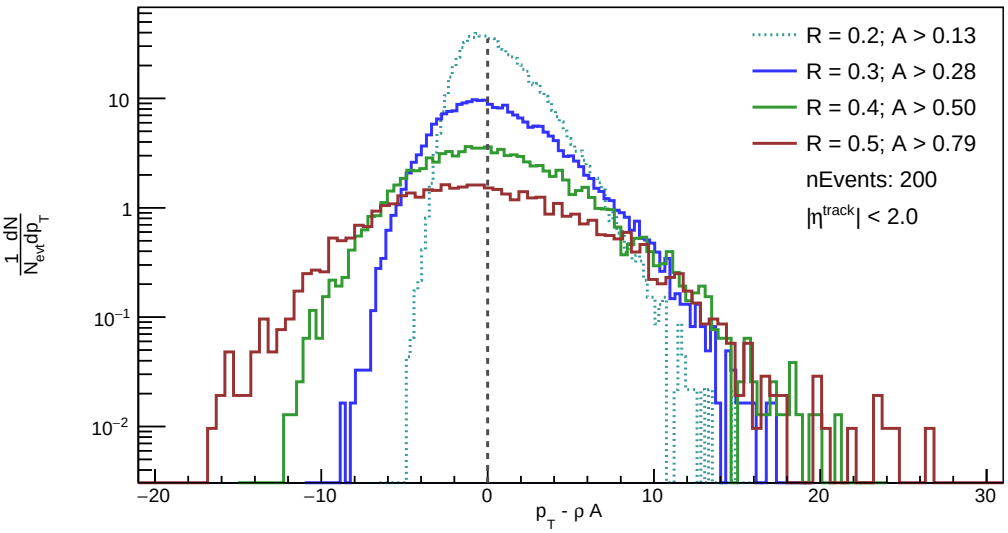

R = 0.2; A > 0.13
R = 0.3; A > 0.28
R = 0.4; A > 0.50
R = 0.5; A > 0.79
nEvents: 2000
|η^track| < 0.9
Counts
ρ


R = 0.2
R = 0.3
R = 0.4
R = 0.5
Num Events: 2000
Jet Area > 0.15
η


R = 0.2; A > 0.13
R = 0.3; A > 0.28
R = 0.4; A > 0.50
R = 0.5; A > 0.79
nEvents: 200
|η^track| < 2.0
Counts
ρ


π(0.2)²
π(0.3)²
π(0.4)²
π(0.5)²
R = 0.2; A > 0.13
R = 0.3; A > 0.28
R = 0.4; A > 0.50
R = 0.5; A > 0.79
nEvents: 200
|η^track| < 2.0
1/N_evt dN/dA
Jet Area


R = 0.2; A > 0.13
R = 0.3; A > 0.28
R = 0.4; A > 0.50
R = 0.5; A > 0.79
nEvents: 200
|η^track| < 2.0
Counts
Num Jets per Event


R = 0.2; A > 0.13
R = 0.3; A > 0.28
R = 0.4; A > 0.50
R = 0.5; A > 0.79
nEvents: 200
|η^track| < 2.0
1/N_evt dN/dp_T
p_T - ρ A