Note of Monte Carlo and Jet Tutorial

Min Zhong

March 23th, 2019

Review of the tutorial

➢ Monte Carlo Simulation

Two common methods are used to generate a list of random numbers with from a list of evenly distributed random numbers

➢ Accept-Reject Method

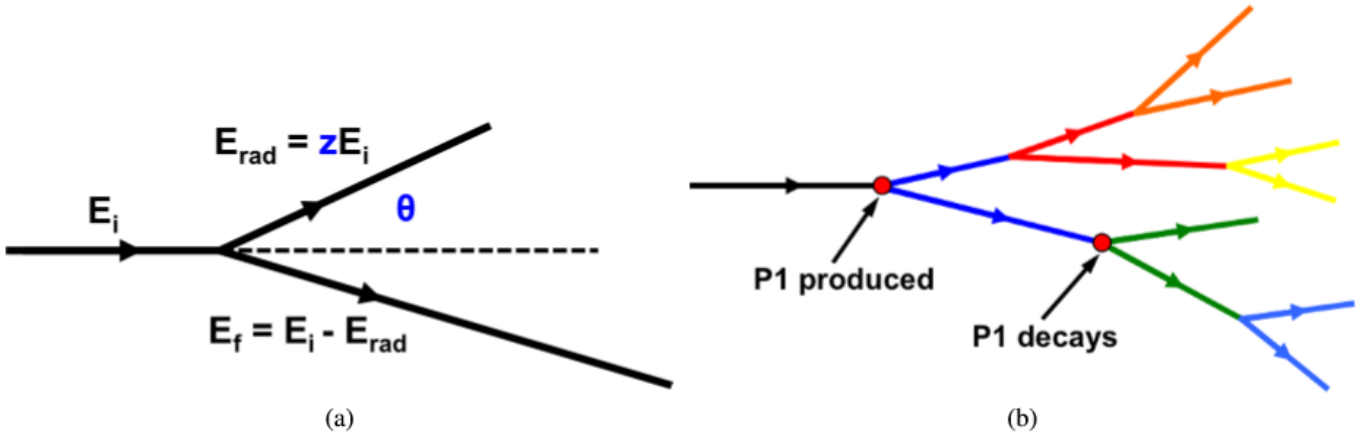
Firstly get a list of uniform random numbers $\{x\}$, another list of evenly distributed random number $\{y\}$ is used. Whether a x will be accept or not is determined by the comparison of y and $PDF(x)$

➢ Inverse Transform Method

Firstly get a list of uniform random numbers $\{u\}$, then a x value is gotten by : $x = CDF^{-1}(u)$, where $CDF(x)$ is the cumulative density function of x

Review of the tutorial

➢ Parton Shower Generation



A single one-to-two splitting (left) highlighting the stochastic observables in the process and the composition of multiple splittings to form a full parton shower (right).

- z: we set $PDF(z) = \frac{1}{1+z}$ to avoid the singularity
- θ: we set $PDF(\theta) = \frac{1}{\theta}$ to keep that splitting are preferentially at small angles, "collinear". But θ is limited in $[0.001, \pi/2]$ here.
- For a 3-D parton shower, another dimension φ is added here, and $PDF(\varphi) = 1/2\pi$

Review of the tutorial

➢ Jet Reconstruction

➢ Jet clustering algorithm

$$d_{ij} = min(p_{Ti}^{2n}, p_{Tj}^{2n}) \times \frac{\Delta_{ij}}{R}$$

$$d_{iB} = p_{Ti}^{2p}$$

- Two observables are defined here
  the distance between two pseudojets: d_{ij}
  the distance between a pseudojet and the beam: d_{iB}
- Calculate d_{ij} matrix and d_{iB} matrix
- Find the minimum value of the two matrixes
- If the minimum value is a d_{iB}, pick out the corresponding particles to become a real jet
- If the minimum value is a d_{ij} , combine the particles i and j to reconstruct a new pseudojet and recalculate the matrixes
- Continue the process utill all pseudojets are reconstruced to jets

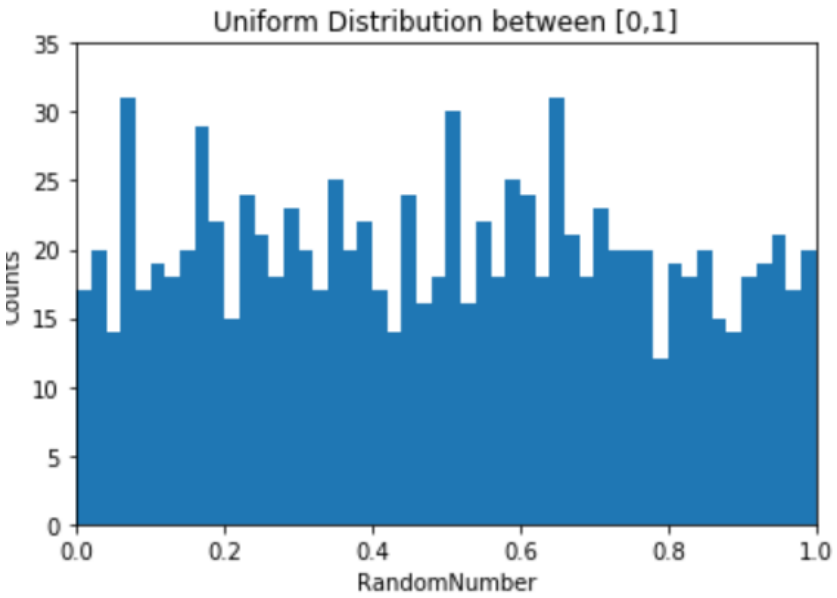
Exercise and Results

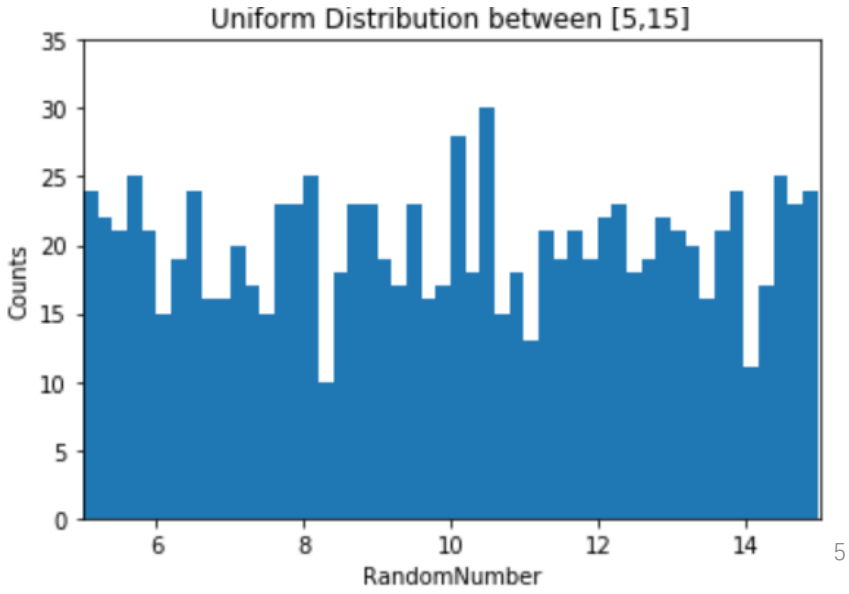
➢ Problem Description

Practice writing a program in PYTHON that generates random numbers. In the first case, write a program to generate 1000 random numbers distributed uniformly between [0,1] and make a histogram of these numbers. In the second case, write a program that distributes the random numbers between [5,15] and makes a histogram. Remember, it is only allowed to use the `random.random()` functionality in PYTHON. As a hint, think about how to make the minimum and maximum of the [0,1] distribution into the minimum and maximum of the [5,15] distribution.

➢ Result

➢ The random number distributed uniformly between [0,1] and [5,15] are shown below





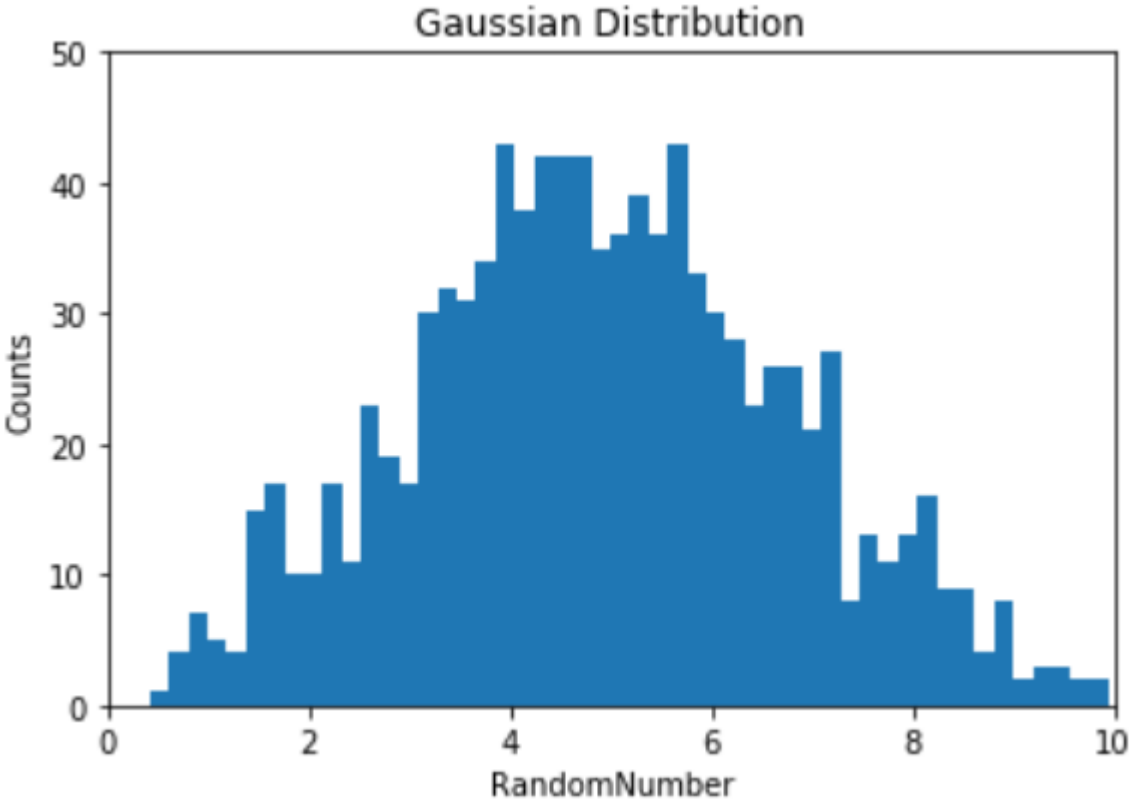
Exercise and Results

➢ Problem Description

Practice writing a program in PYTHON that generates random numbers, distributed according to a Gaussian distribution with a mean μ of 5 and a width σ of 2 in the range of [0,10].

➢ Result

- Accept-Reject method is used here, and the result is shown below



Exercise and Results

➢ Problem Description

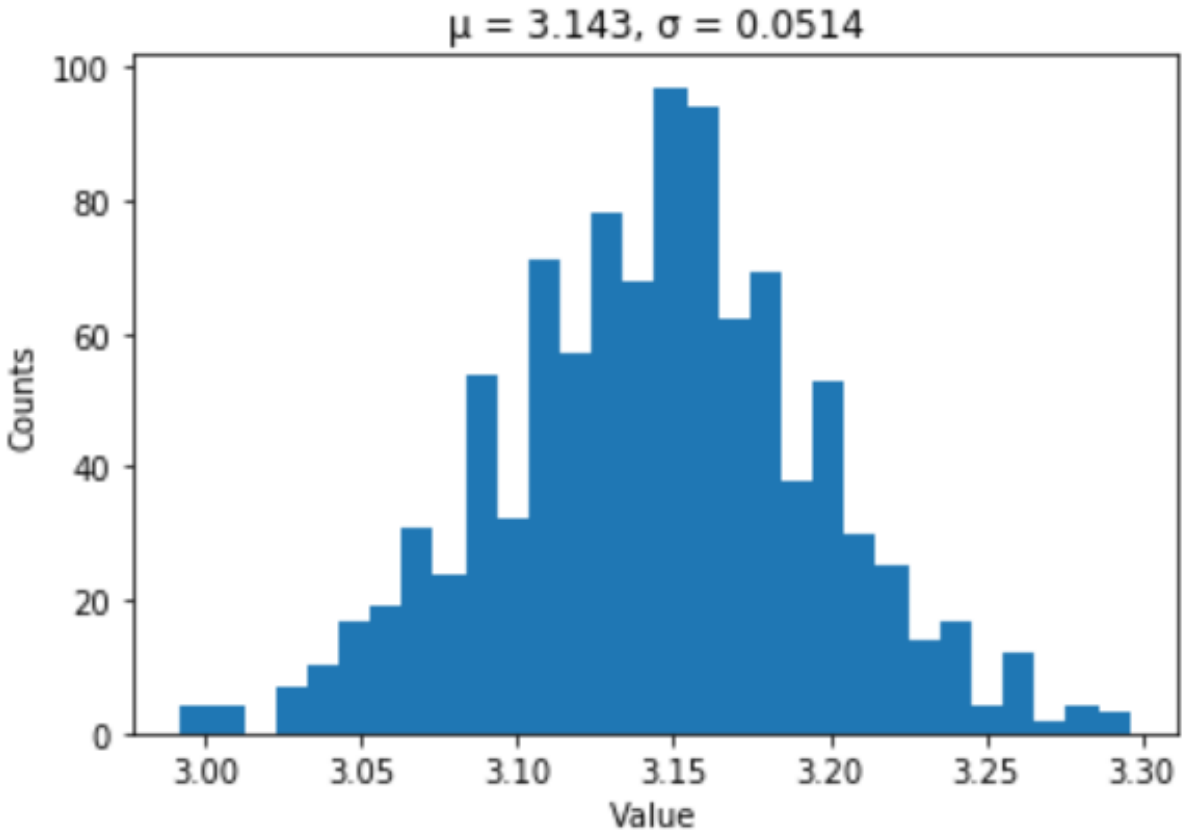
Use accept-reject method to calculate the value of π. The raw description is very long, so I divide the problem into three parts here

- Part 1: Calculate π for 1000 times (every time with 1000 points) and get the mean μ and standard deviation σ
- Part 2: Increase the times of calculation to see how σ changes
- Part3 : Increase the number of points to see how σ changes

Exercise and Results

➢ Result

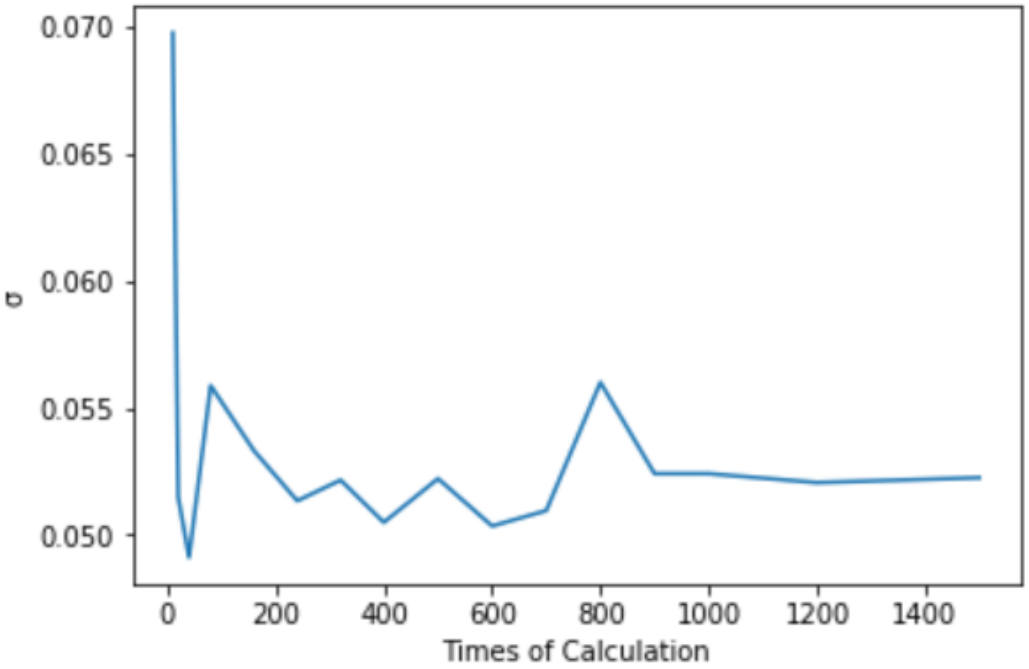
➢ Part 1: Calculate π for 1000 times (every time with 1000 points) and get the mean μ and standard deviation σ



Exercise and Results

➢ Result

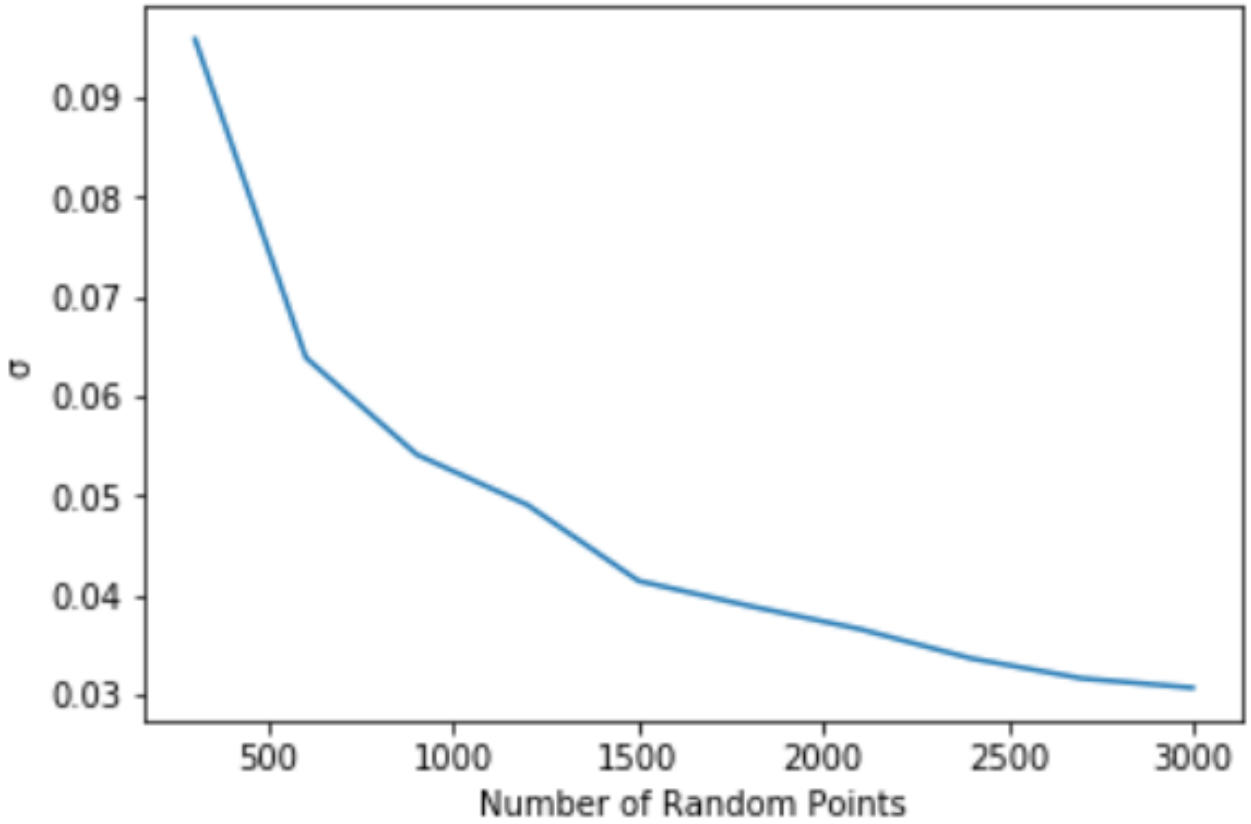
- Part 2: Increase the times of calculation to see how σ changes (every time with 1000 points)



- As calculation times become larger, σ becomes stable. I think it is because that the distribution of the value calculated is a **certain Gaussian distribution (μ , σ)** , and in fact we are **estimating σ** of the distribution here

Exercise and Results

- **Result**
  - Part3 : Increase the number of points to see how σ changes (Calculation times remains 1000)



  - The number of random points has a great influence on σ, as the number increases, σ decreases significantly

Exercise and Results

➢ Problem Description

Practice writing a program in PYTHON that generates random numbers, distributed according to a falling distribution of the form e^{-x} in the range [0,5]. After doing this, generate the same distribution using the accept-reject method and determine the efficiency of the generation, defined as the fraction of accepted x values with respect to the total generated x values. How does this efficiency compare for the accept-reject method and the inverse transform method?

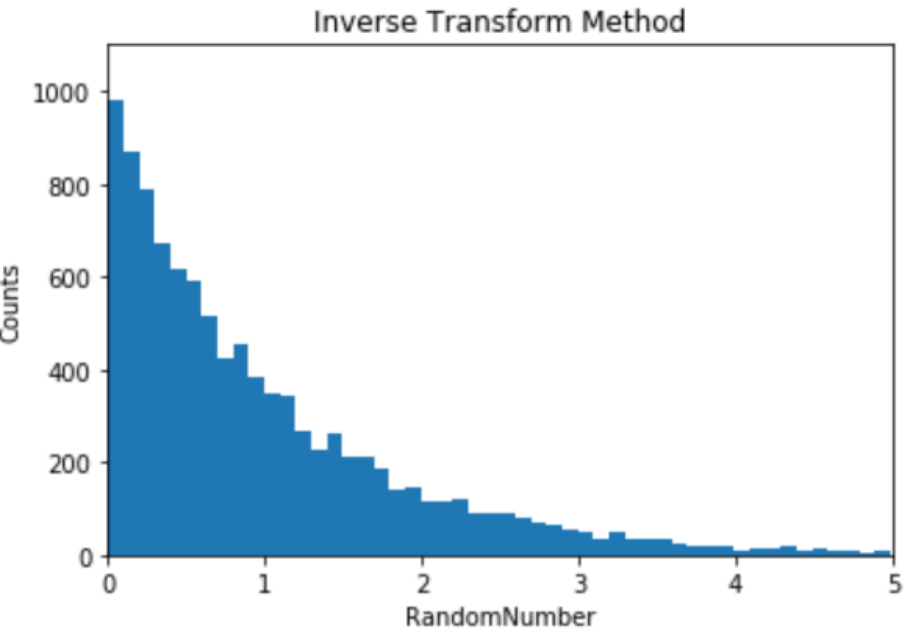
➢ Result

- Accept-Reject method and Inverse Transform method are used here, and the result is shown below (generate both 10000 times)

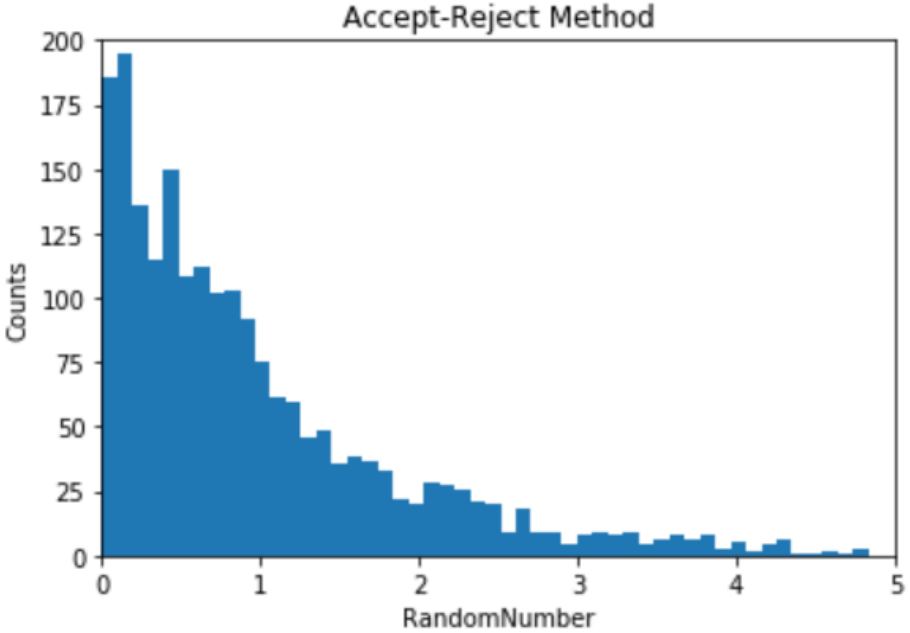
- Two histograms are shown next page

Exercise and Results

➢ Result



Efficiency:0.9939



Efficiency:0.2041

- We can see that the efficiency of Inverse Transform Method is much larger than that of Accept-Reject Method

- Note that we skip some random numbers which are larger than 5, so the efficiency of Inverse Transform Method is not perfectly 100%

Exercise and Results

➢ Problem Description

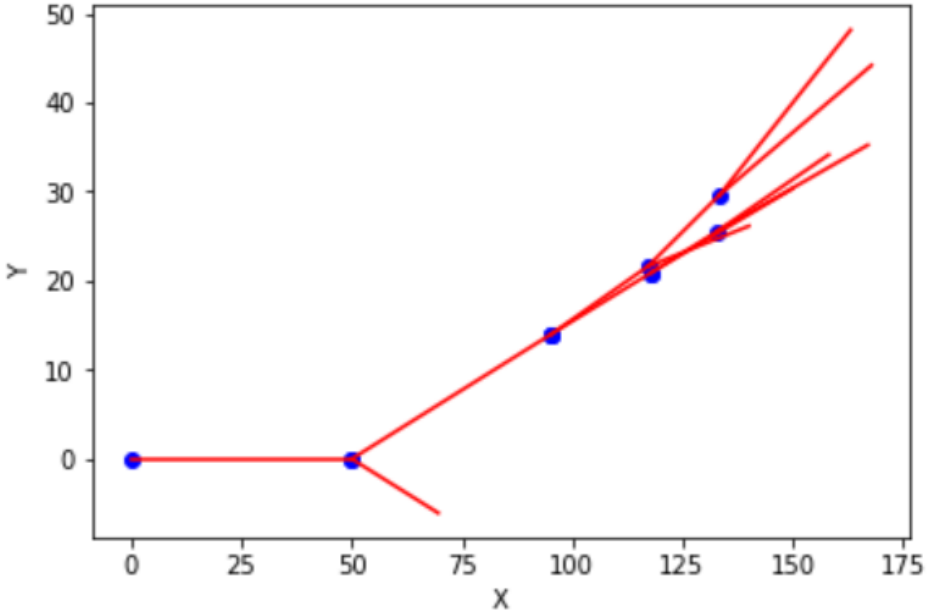
This part is about Parton shower and jet reconstruction, and the first exercise here is to write a program to generate a 2-D Parton shower

➢ Result

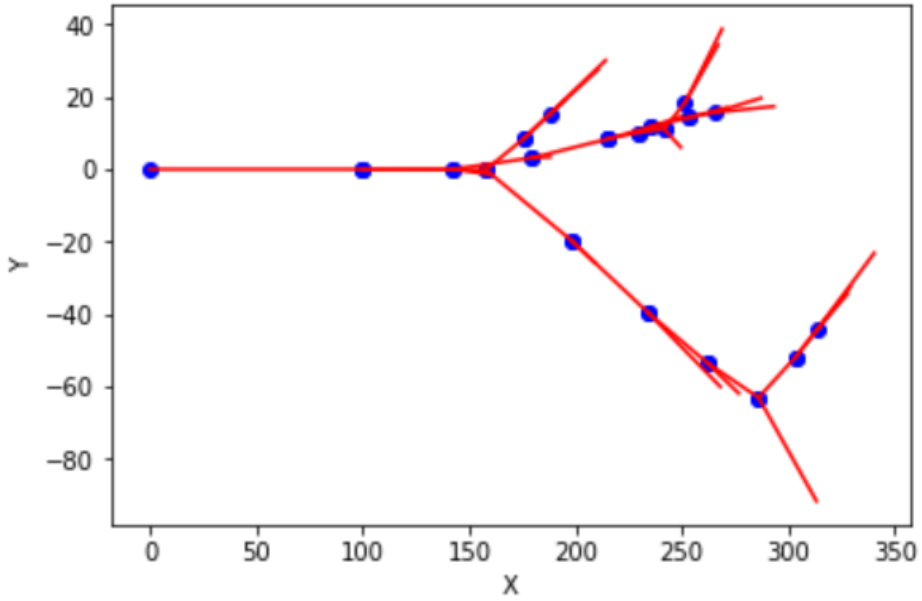
- A simplifying assumption is that for all the particles here, the energy scale $E >> m_0$, thus we can neglect m_0, then we have $E \approx p$
- z: we set $PDF(z) = \frac{1}{1+z}$ to avoid the singularity
- θ: we set $PDF(\theta) = \frac{1}{\theta}$ to keep that splitting are preferentially at small angles, "collinear". But θ is limited in $[0.001, \pi/2]$ here
- Input Energy is changeable and E_{crit} is set to 10 GeV
- Two histograms are shown next page

Exercise and Results

➢ Result



$E_0 = 100\ GeV$



$E_0 = 200\ GeV$

➢ The larger E_0 is, the more final state particles are produced

Exercise and Results

➢ Problem Description

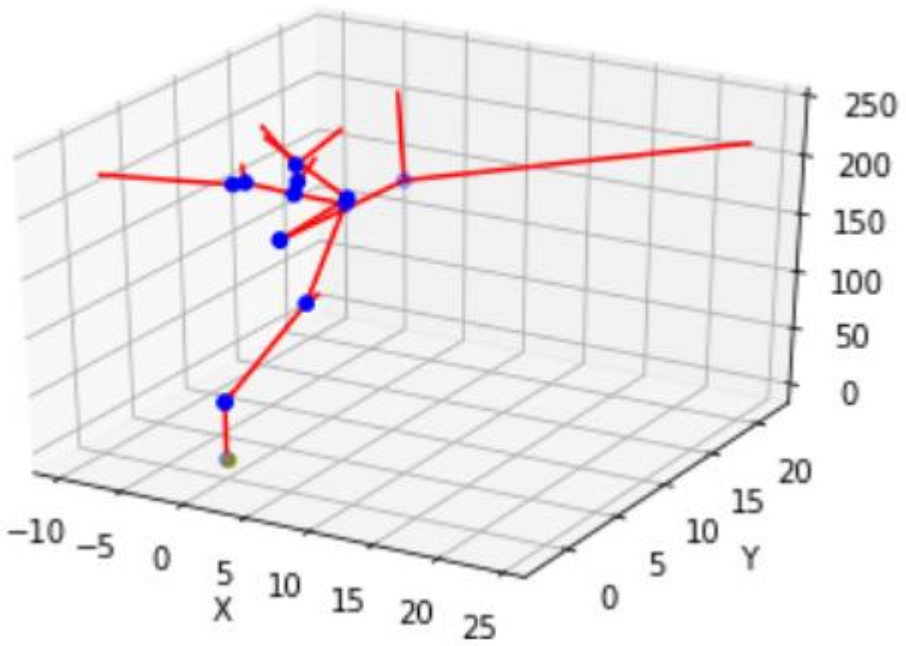
Then we move up to generate a 3-D Parton shower by adding a dimension φ

➢ Result

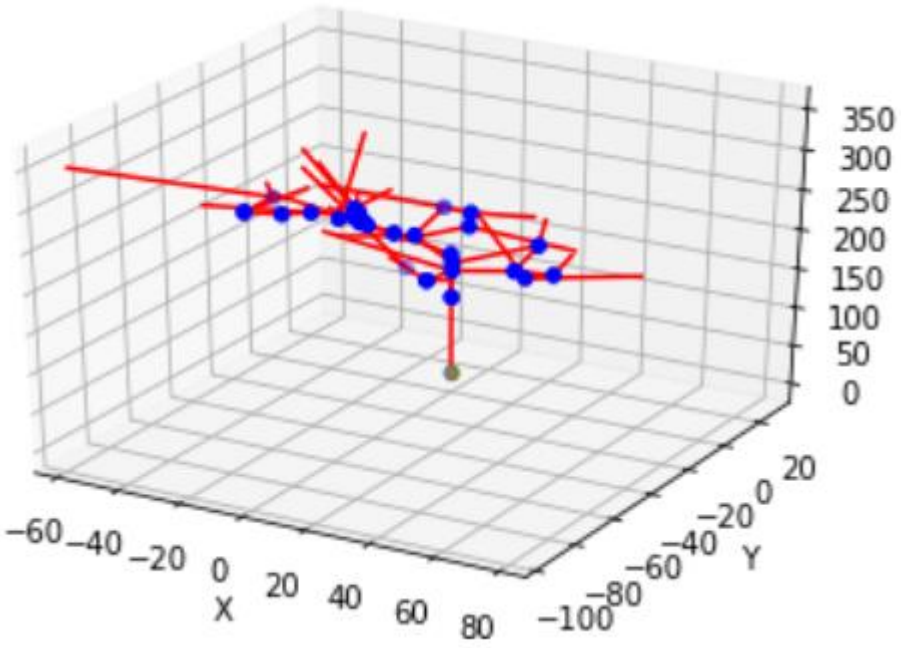
- A simplifying assumption is that for all the particles here, the energy scale $E >> m_0$, thus we can neglect m_0, then we have $E \approx p$
- z: we set $PDF(z) = \frac{1}{1+z}$ to avoid the singularity
- θ: we set $PDF(\theta) = \frac{1}{\theta}$ to keep that splitting are preferentially at small angles, "collinear". But θ is limited in $[0.001, \pi/2]$ here
- $\varphi : PDF(\varphi) = 1/2\pi$, which is a uniform distribution
- Input Energy is changeable and E_{crit} is set to 20 GeV
- Two histograms are shown next page

Exercise and Results

➢ Result



$E_0 = 100\ GeV$



$E_0 = 200\ GeV$

➢ The larger E_0 is, the more final state particles are produced

Exercise and Results

➢ Problem Description

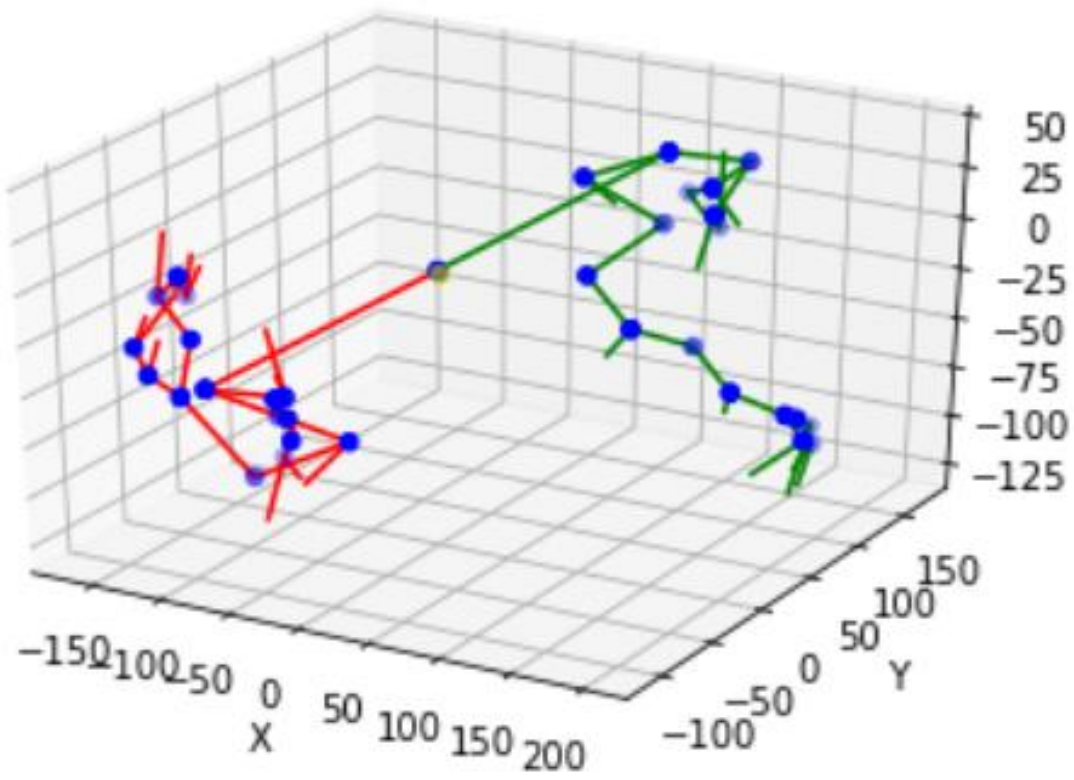
A jet event is generated in this part of exercise

➢ Result

- The initial energy of the two initial state particles which are in opposite directions is determined by $PDF(E) = e^{-\alpha E}$, where α is set to 0.005
- The direction of the initial particles is determined by two evenly distributed random numbers θ_0 and φ_0
- A highest allowed initial E is set to 500 GeV, to avoid too much calculation in following steps
- E_{crit} is set to 20 GeV

Exercise and Results

- **Result**



Exercise and Results

➢ Problem Description

Here we begin to Jet Reconstruction and Pseudo Mass Calculation

From the previous Monte Carlo model you have created, you have the ability to simulate parton showers/hadronizations that are representative of hadronic activity in LHC collisions. At this stage, implement the clustering algorithm described here to cluster a single event, including proper validation of the procedure. Note that you will have to specify the parameters n and R which are choices of the algorithm. Make a histogram[a] of the number of jets that result from this algorithm for 1000 parton showers and do this for the set of n values of [-1,0,1] and R values of [0.01,0.05,0.1,0.5,1.0]. What differences do you see?
A histogram is a great way to check if your program is doing what it is supposed to do. Think about what this histogram should look like and ask youself "Does it make sense?"

Based on the jets that you have clustered in the previous section, calculate the jet "pseudo-mass" as

$$\text{SimpleMass}(J) = E(j_1) \times E(j_2) \times \Delta R(j_1, j_2)$$

where j_1 and j_2 are the two highest p_T constituents of the jet J. In the case that a given parton shower and clustering produces more than one jet, perform this calculation for the jet of highest energy. Perform this calculation for a large number of parton shower generations and make a histogram of the pseudo-mass for all of these generated showers.

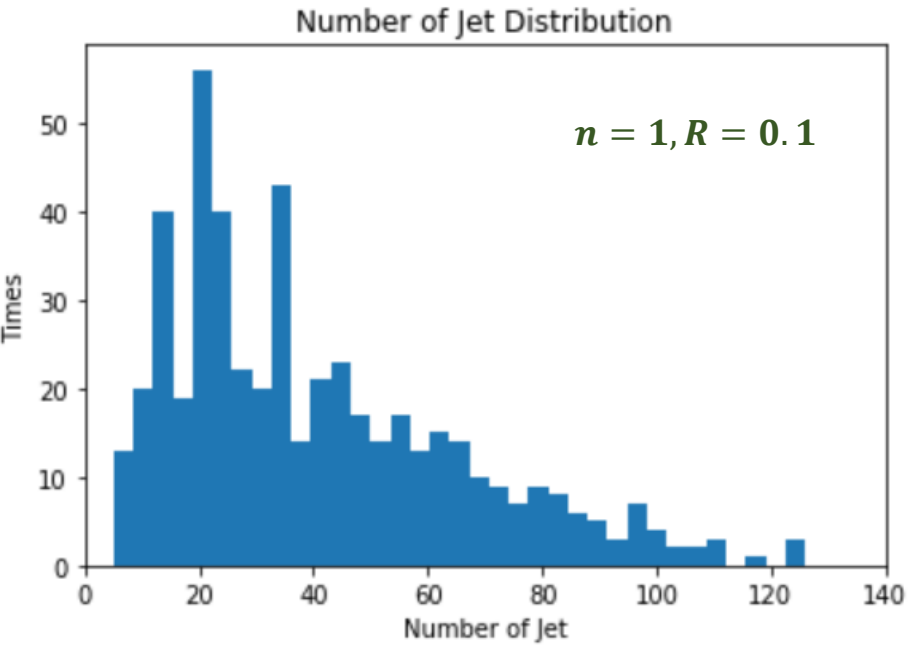
Exercise and Results

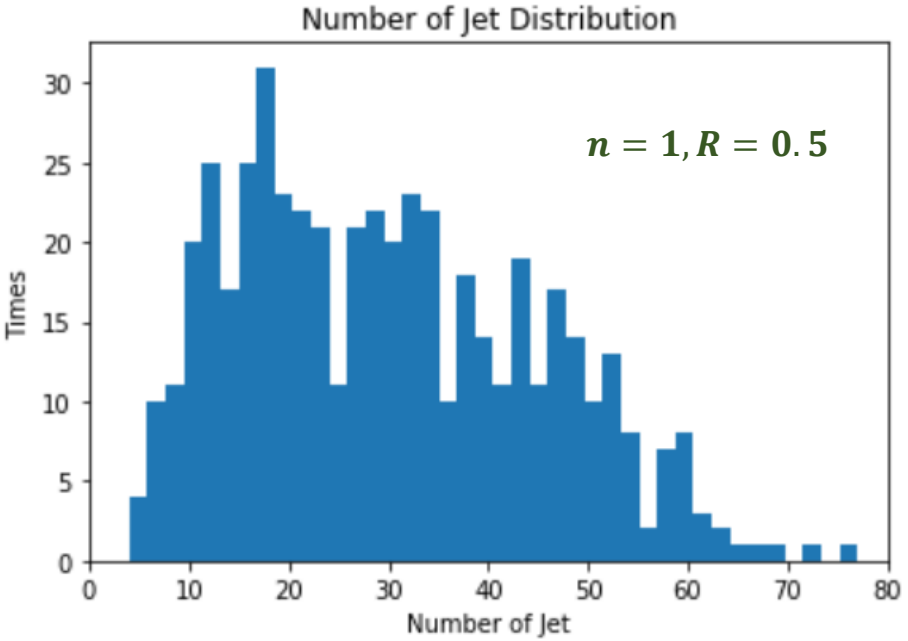
➢ Result

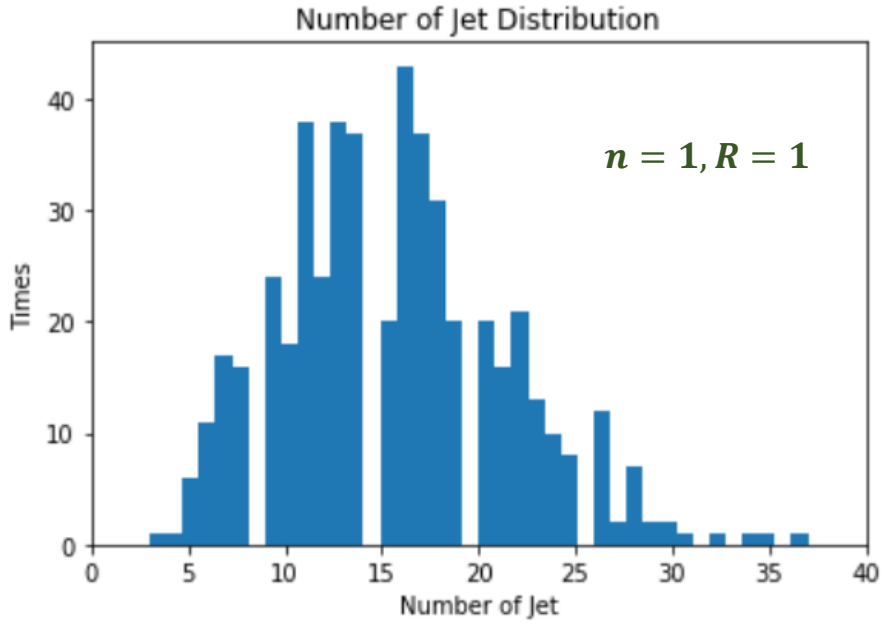
- For each group of $\{n, R\}$ values, 500 Jet Events are generated
- We set $n = \{1, 0, -1\}$, $R = \{0.1, 0.5, 1\}$ here, just to save some time
- Then for each jet event, jet clustering algorithm is applied
- The initial energy of every jet event is determined by $PDF(E) = e^{-\alpha E}$, where α is set to 0.01
- E_{crit} is set to 10 GeV, and the highest permitted E is set to 300 GeV

Exercise and Results

- **Result (Number of Jets)**

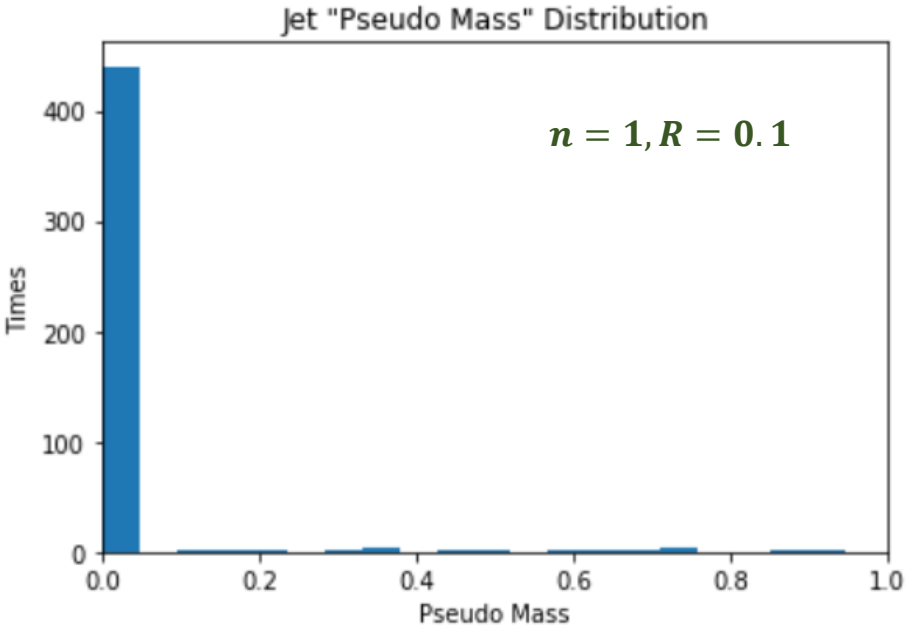


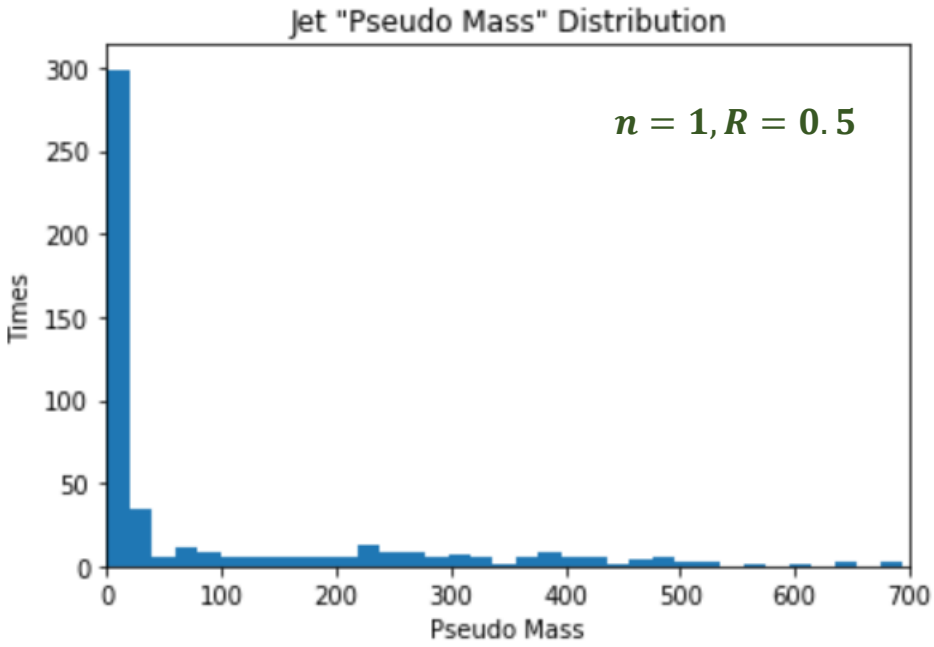


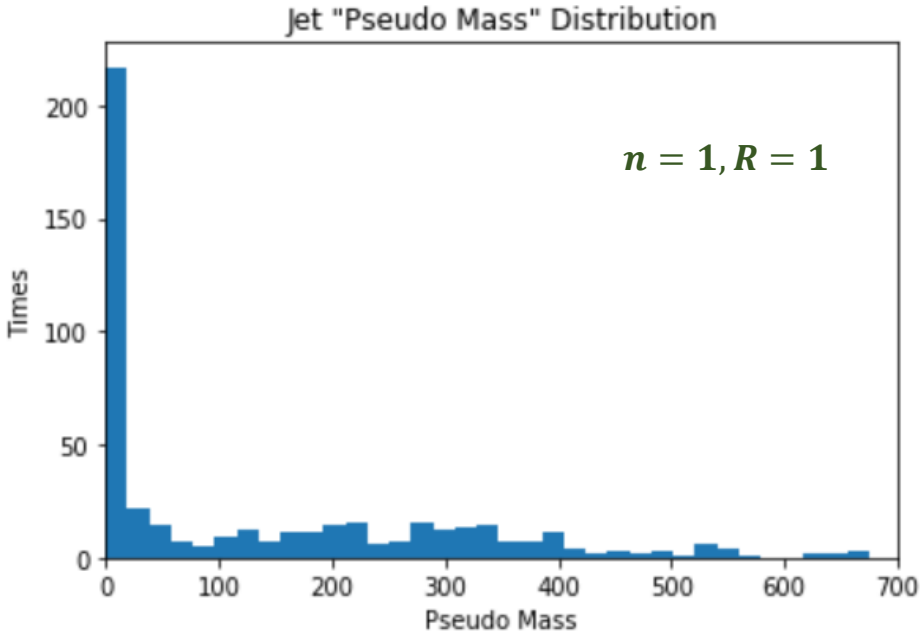


Exercise and Results

- **Result (Pseudo Mass)**

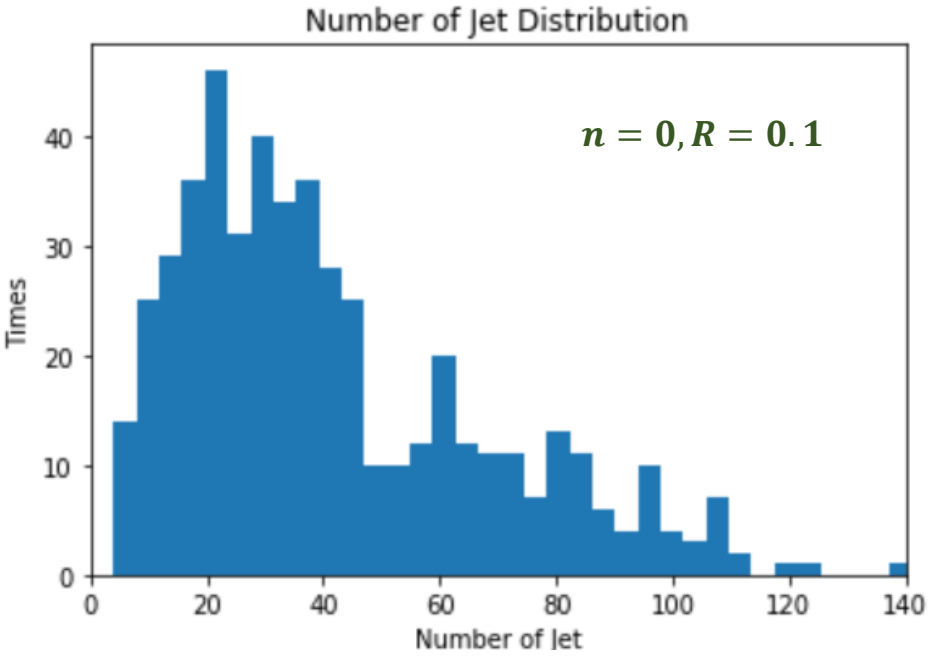


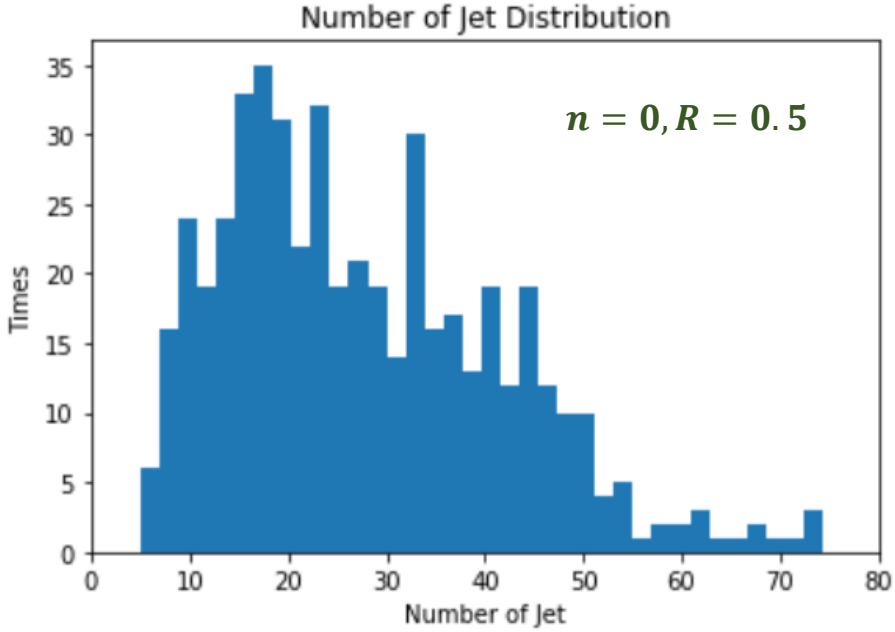


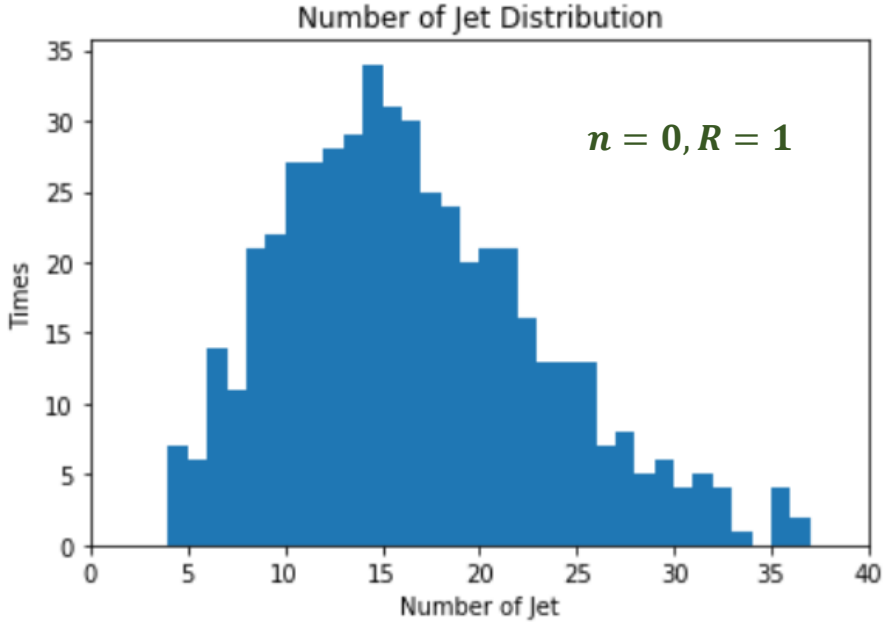


Exercise and Results

- **Result (Number of Jets)**

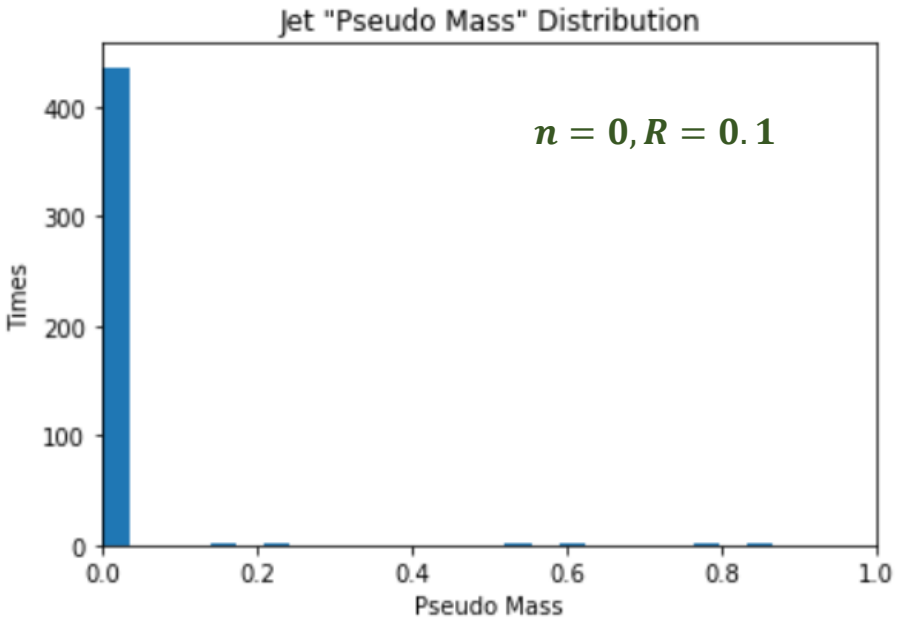


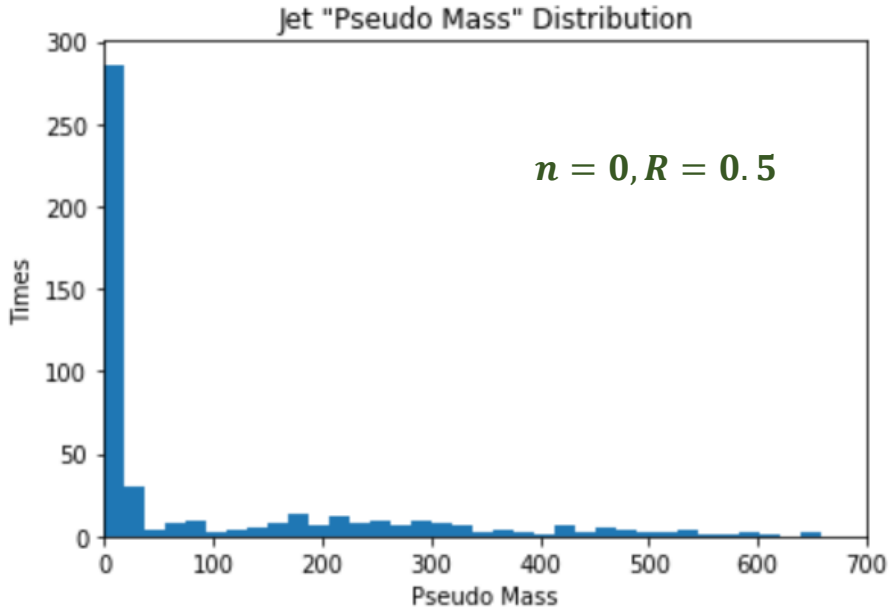


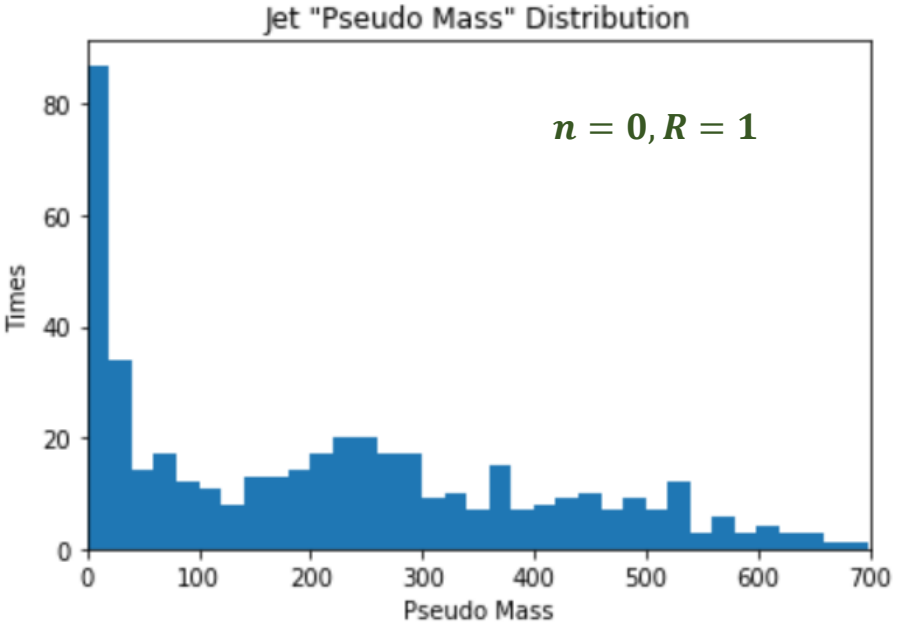


Exercise and Results

- **Result (Pseudo Mass)**

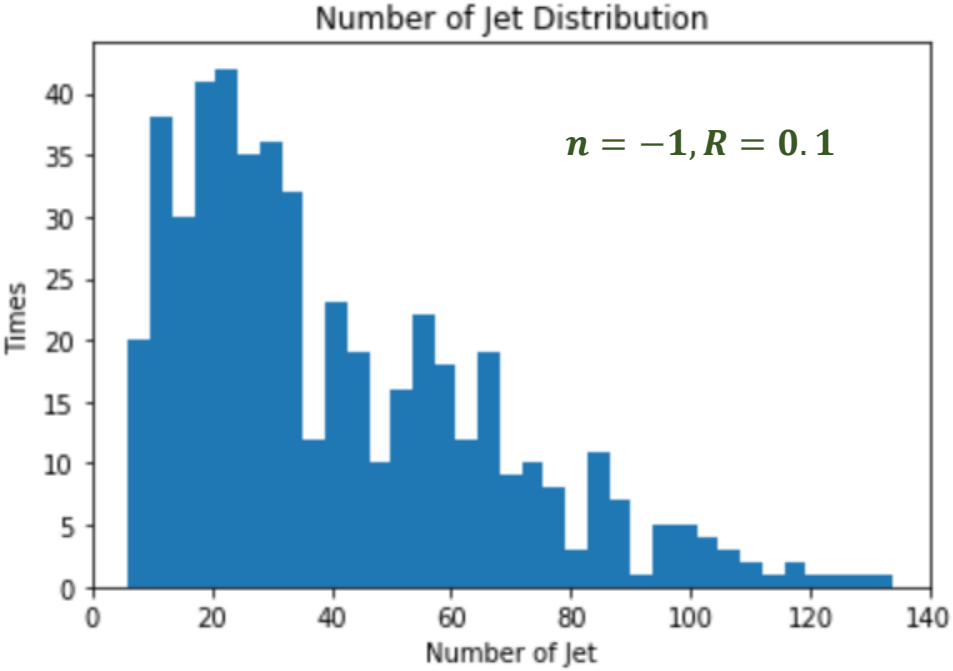


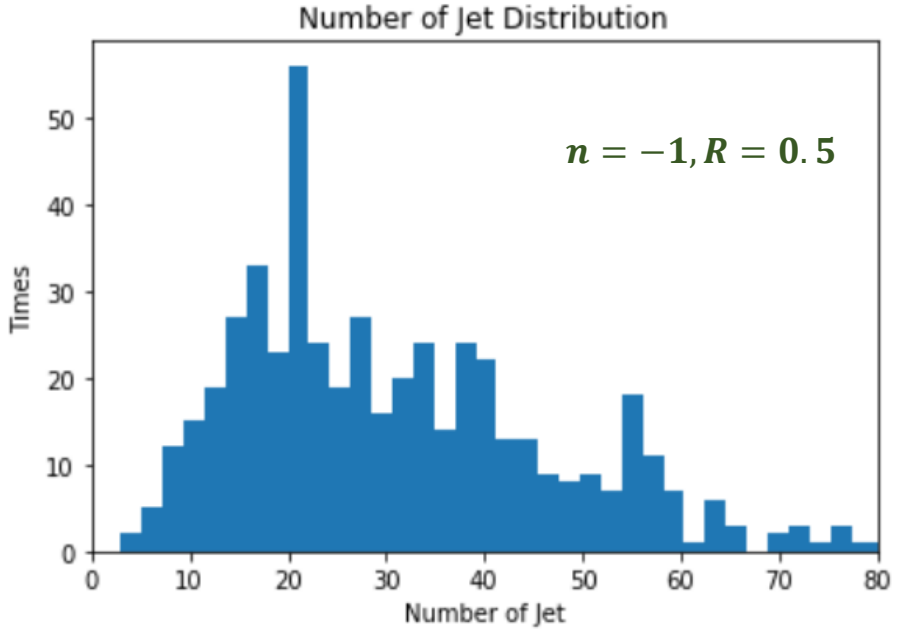


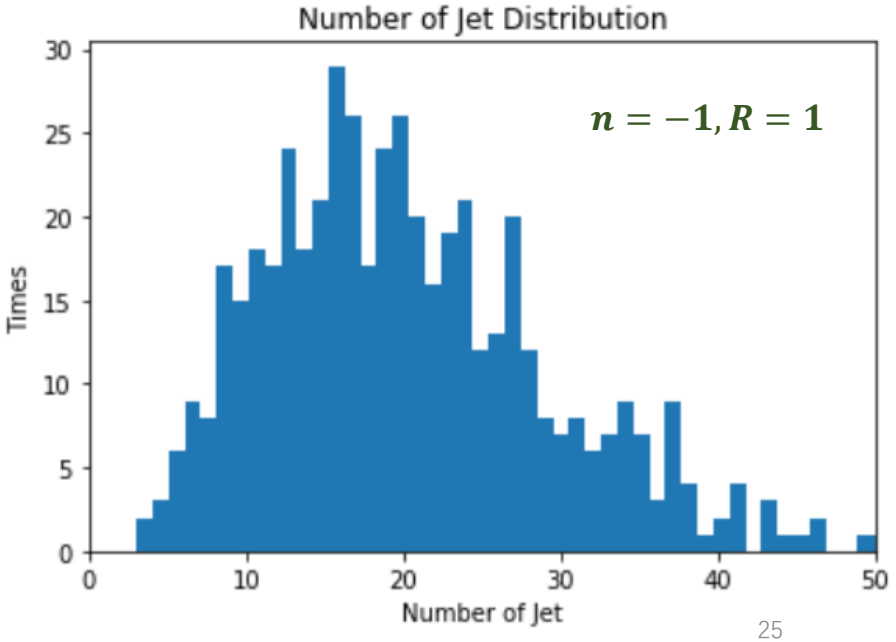


Exercise and Results

- **Result (Number of Jets)**

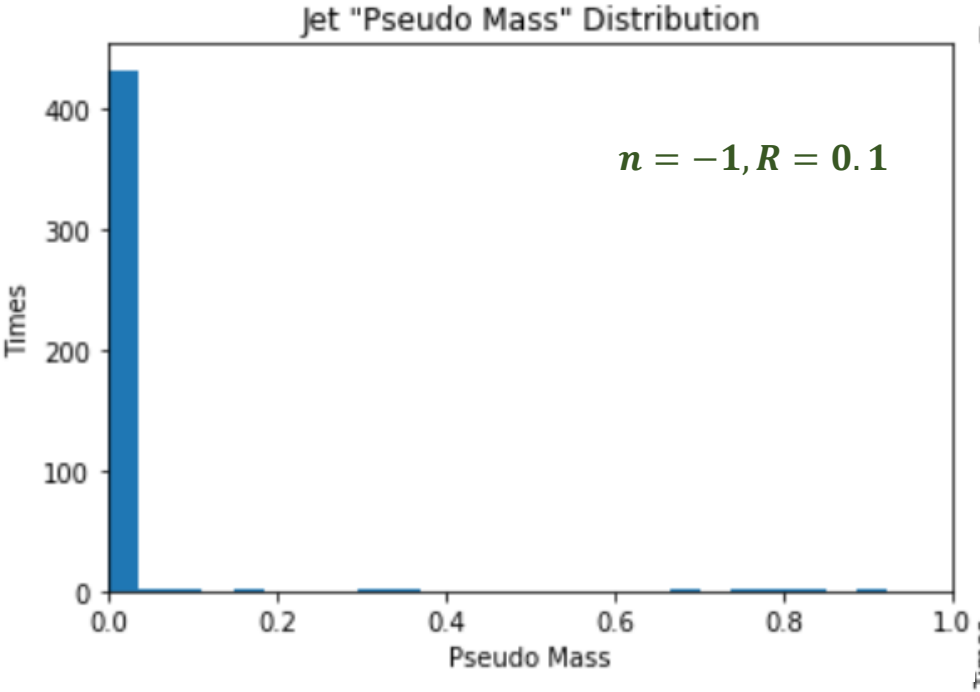


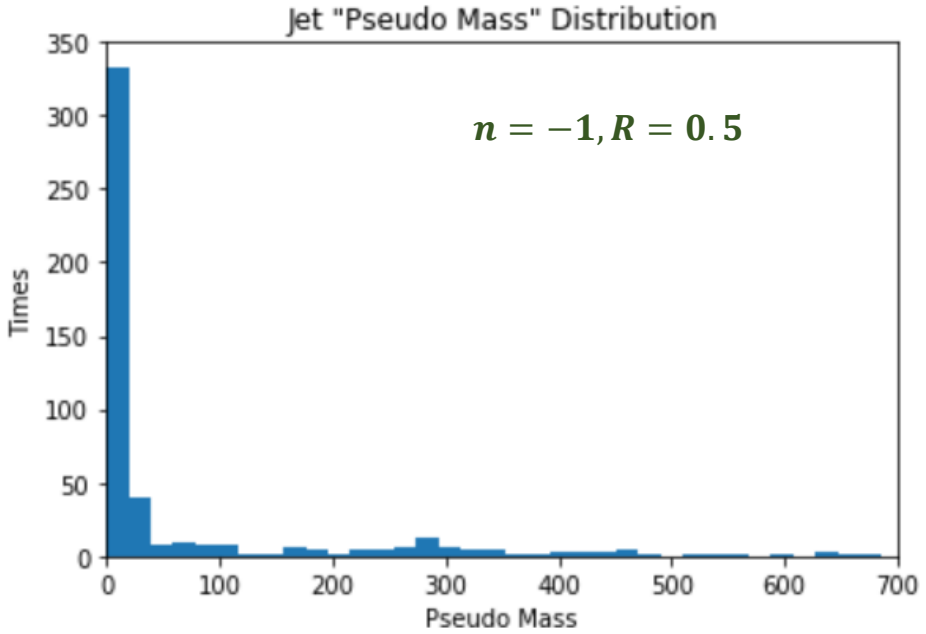


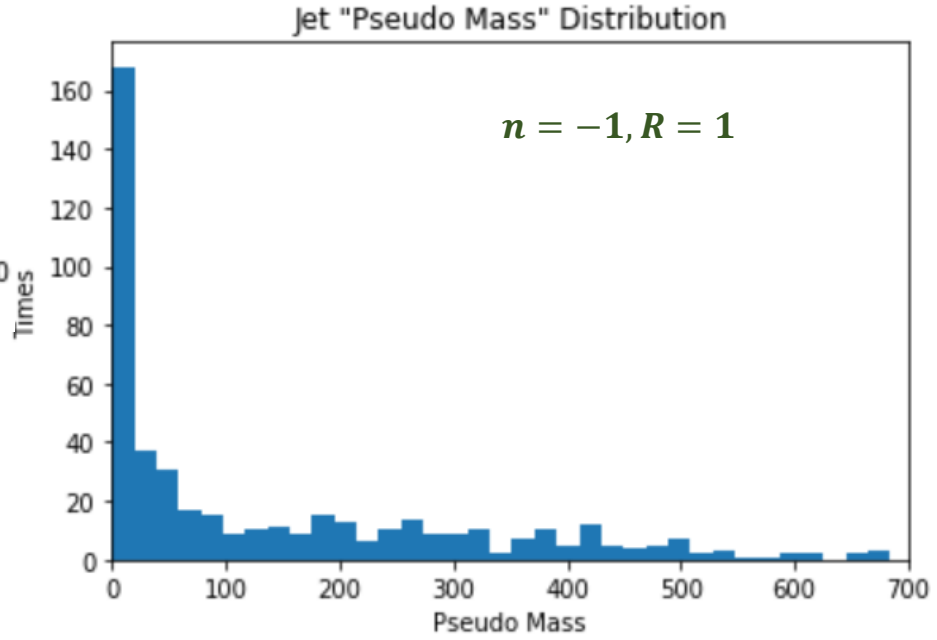


Exercise and Results

- **Result (Pseudo Mass)**







Exercise and Results

➢ Result

- The value of n has small influence on the Number of Jets
- R value has large impact on the Number of Jets, the larger R is , less jets will be reconstructed, and Jet Pseudo Mass is larger
- Note that we have changed the initial Energy of each shower every time ($PDF(E) = e^{-\alpha E}$), so the Number of Jets' distribution is not a simple Gaussian distribution. It is supposed to be **a convolution of Gaussian distribution and exponentially falling distribution.**
- According to my test, if we set the initial Energy to constant, the distribution of Number of Jets is similar to Gaussian distribution, but E_{crit} influences the distribution, too

All the codes can be found at https://github.com/Min-Zhong/Exercise-of-Monte-Carlo-And-Jet-Tutorial