

---

# Looking into Jets with Machine Learning

**Sebastian Macaluso**  
New York University

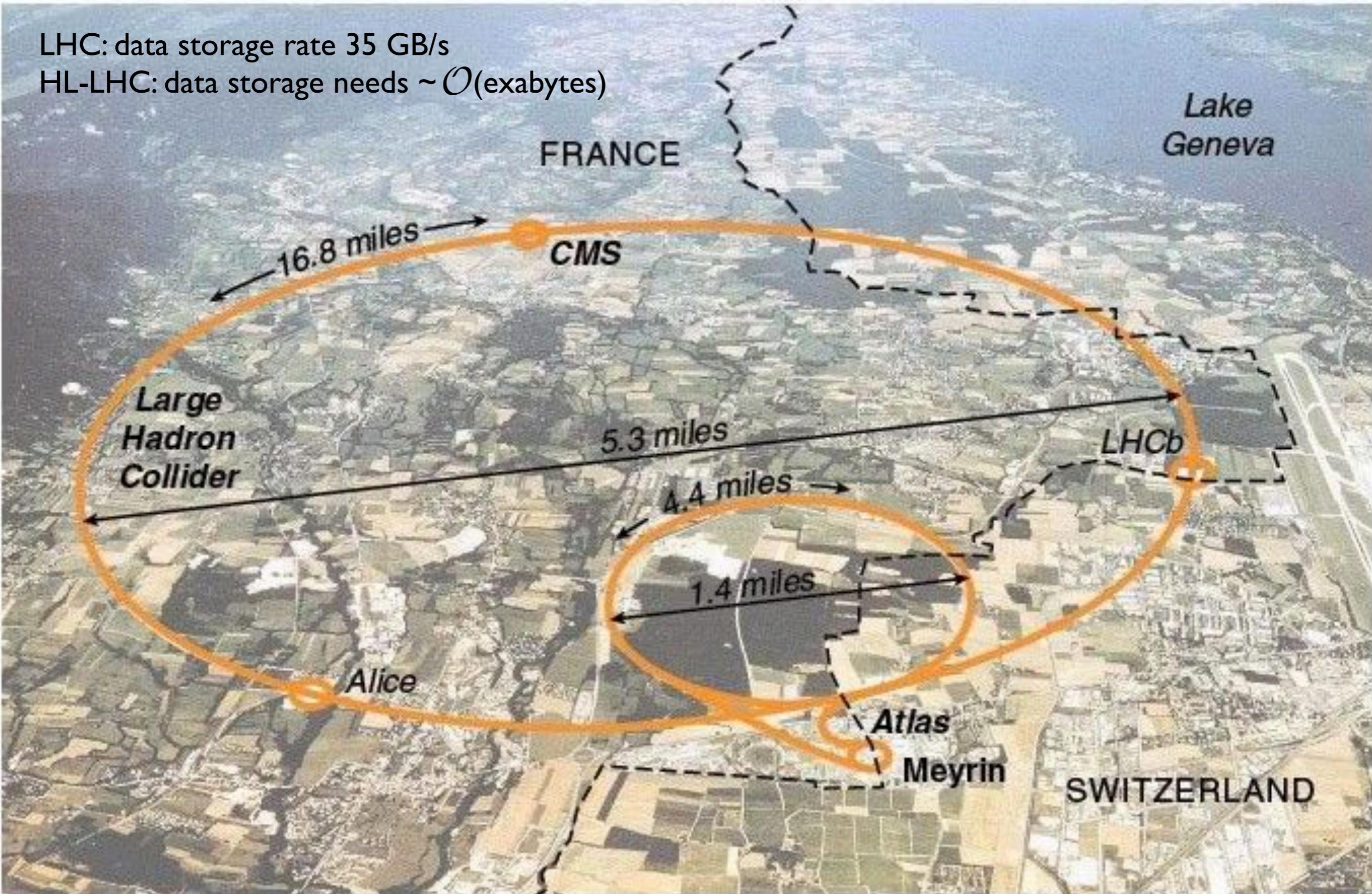
DS Tea  
UMass Amherst  
November 4, 2019



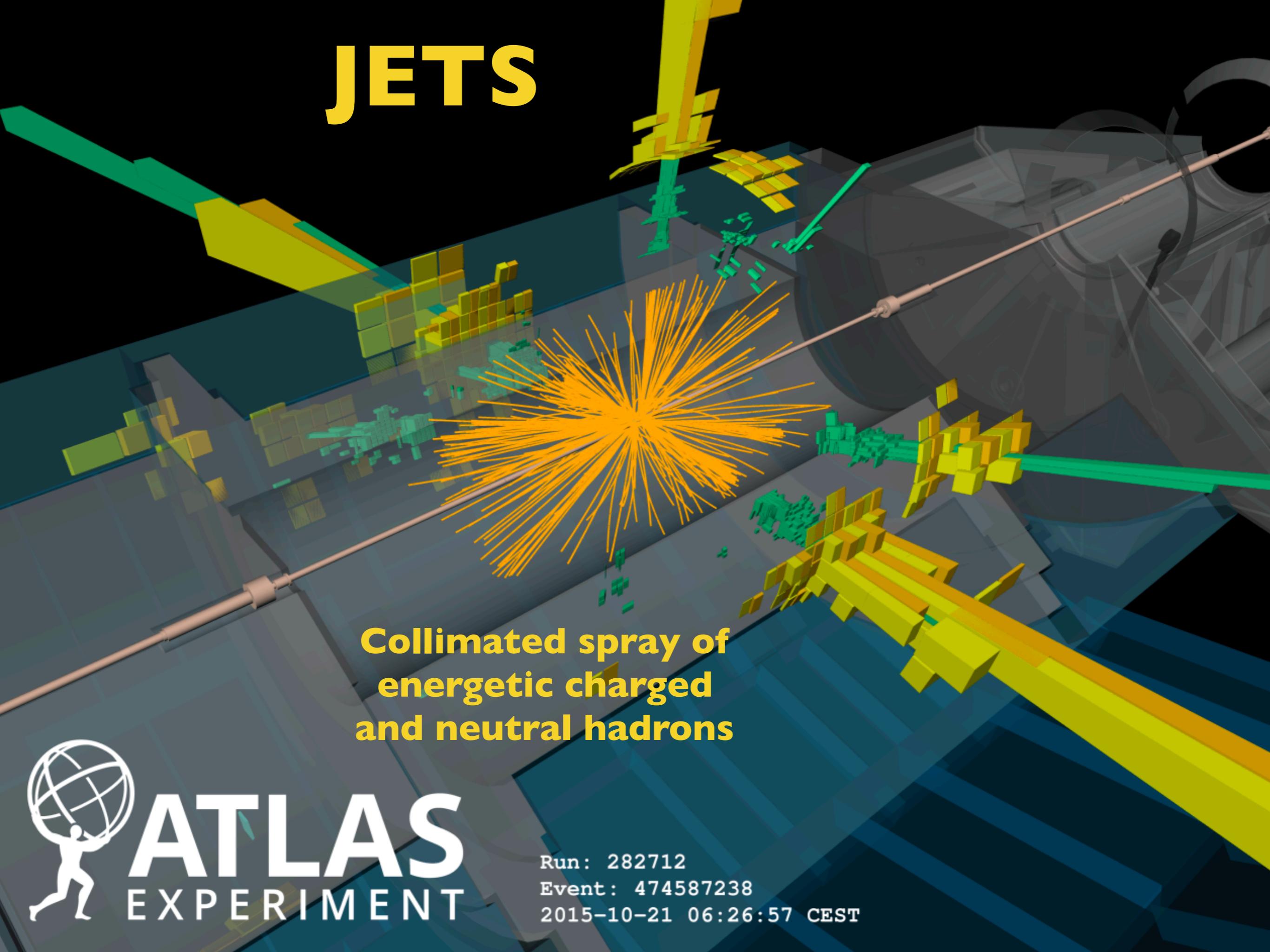
# The Large Hadron Collider

LHC: data storage rate 35 GB/s

HL-LHC: data storage needs  $\sim \mathcal{O}(\text{exabytes})$



# JETS

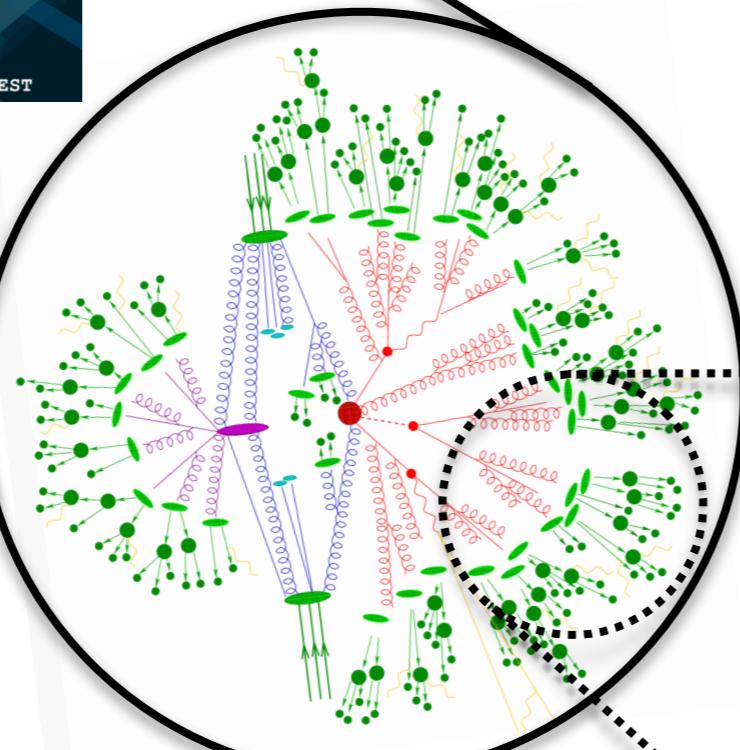
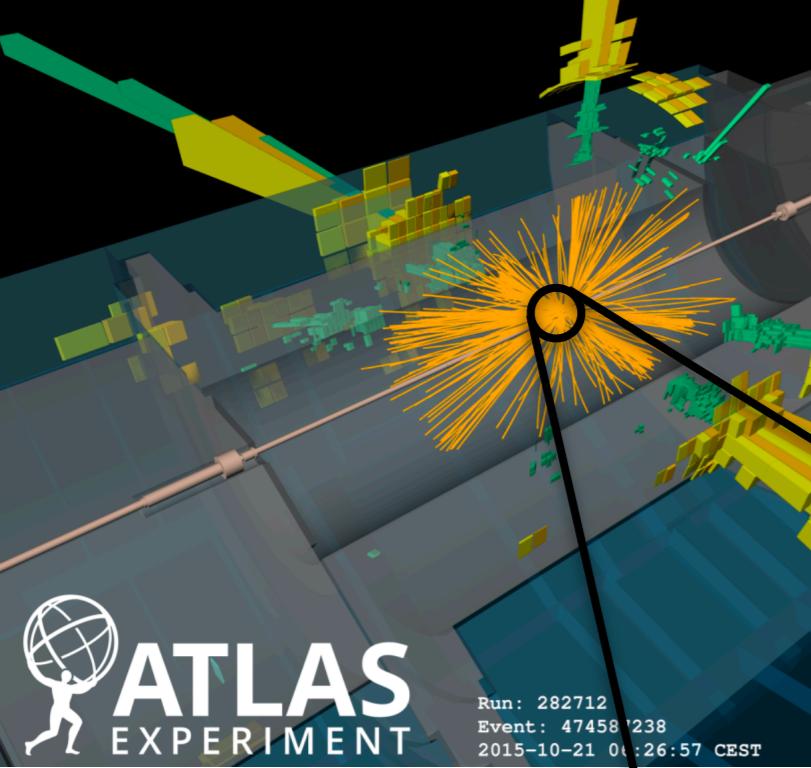


**Collimated spray of  
energetic charged  
and neutral hadrons**

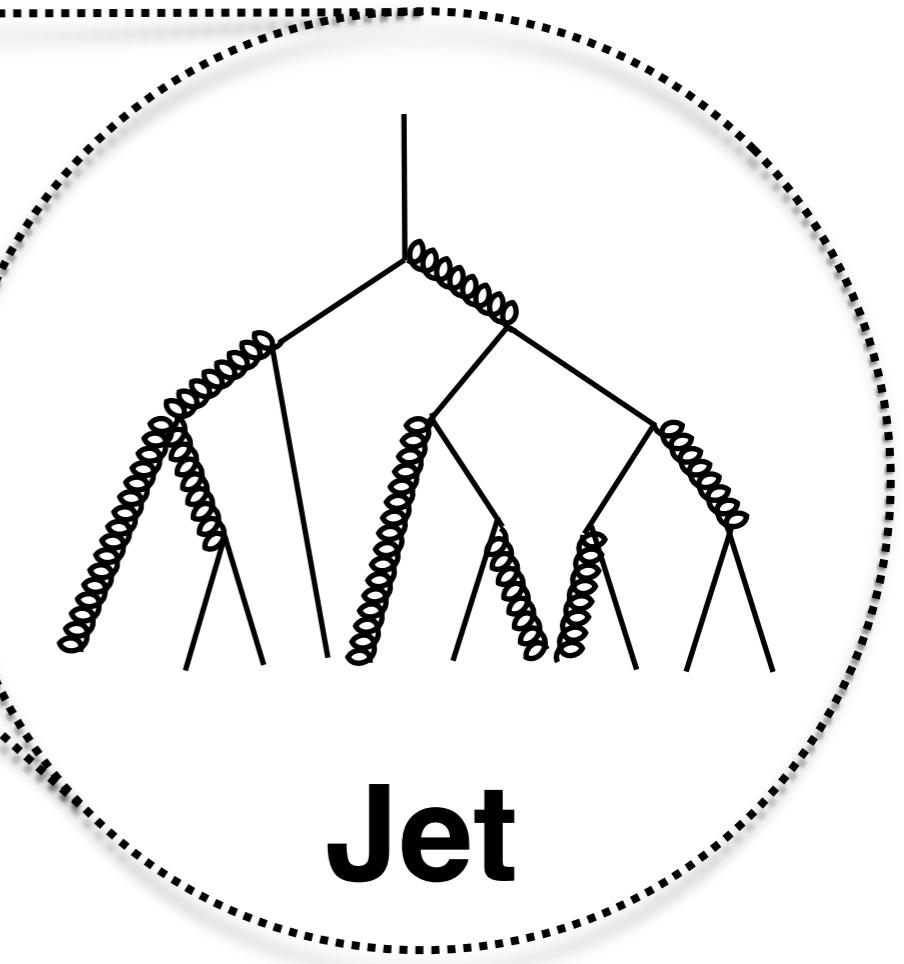


**ATLAS**  
EXPERIMENT

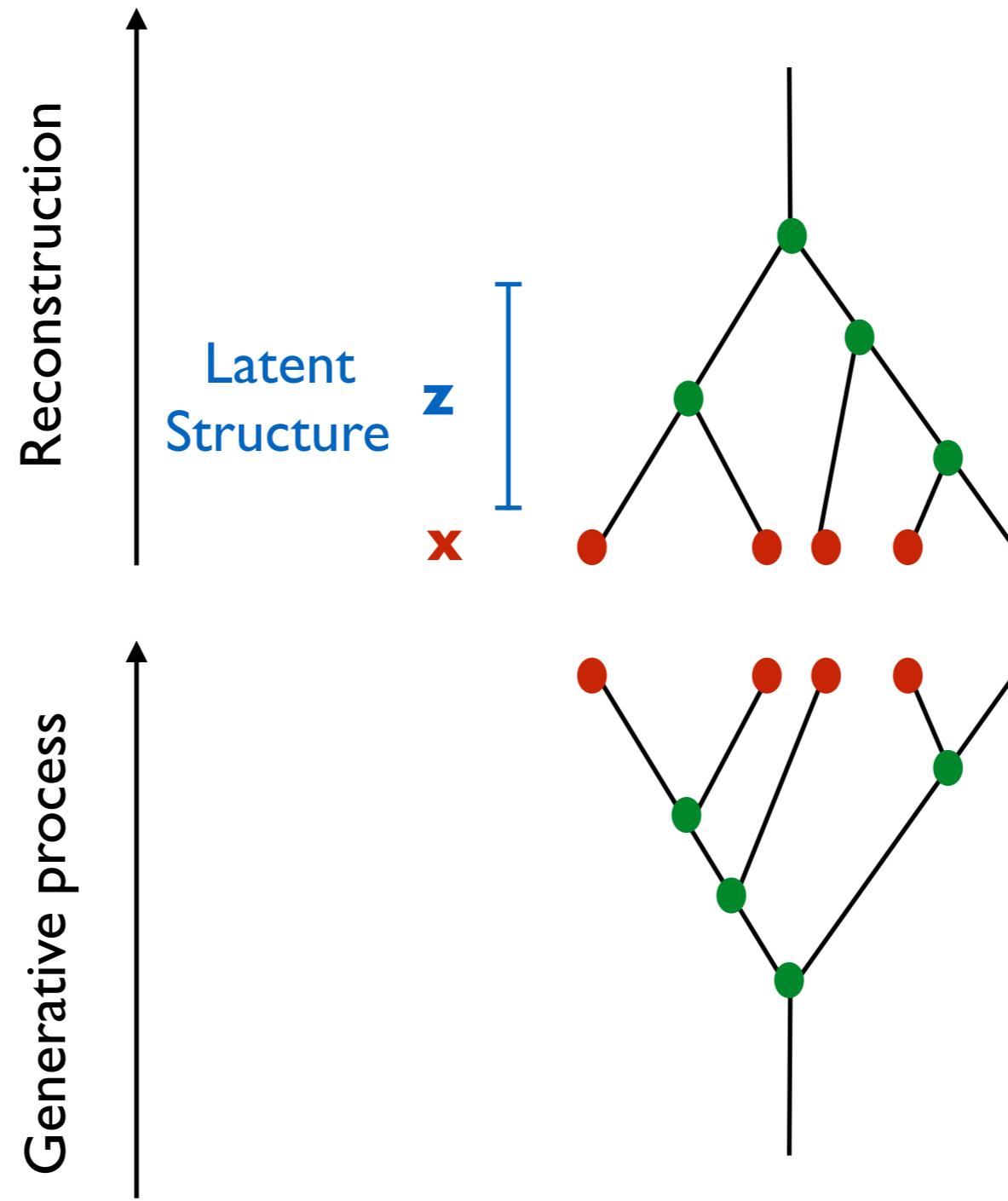
Run: 282712  
Event: 474587238  
2015-10-21 06:26:57 CEST



[Stefan Hoche 1411.4085]



# Jet clustering

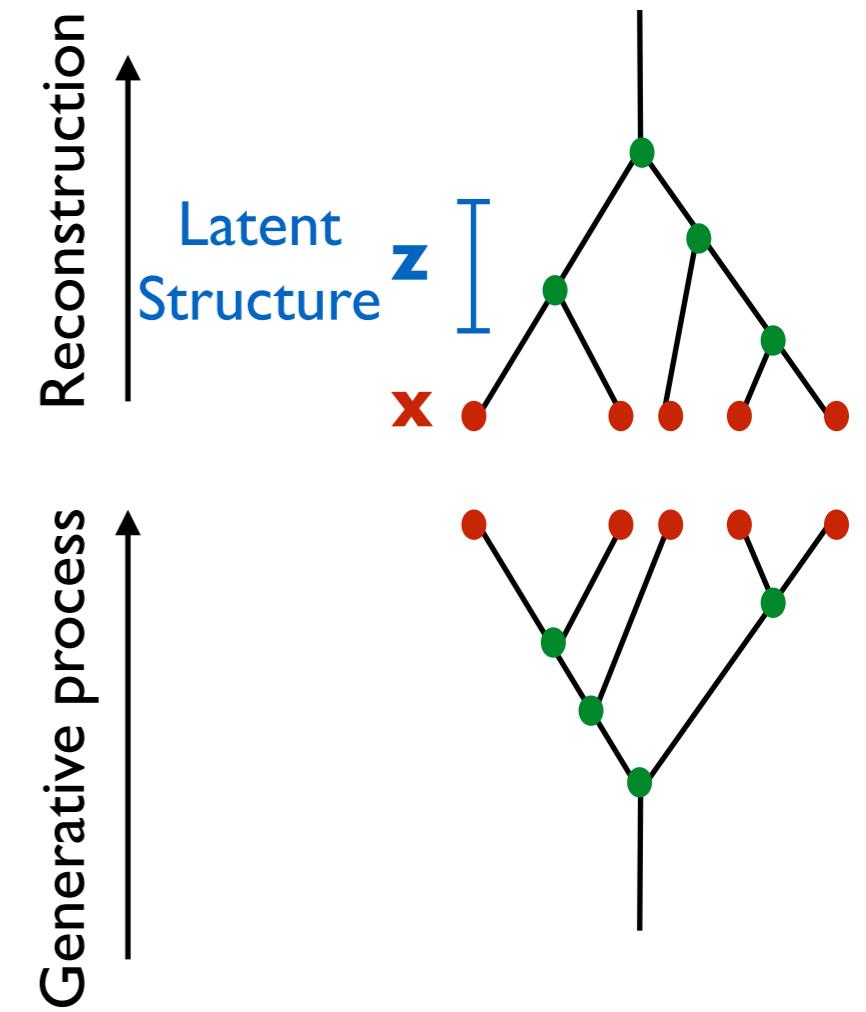


# Problem settings

**Classification**

**Maximum likelihood estimate**

**Posterior distribution on histories**

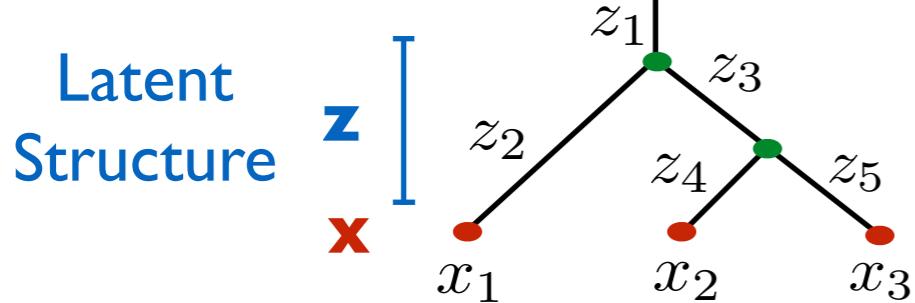


---

# Outline

- **Unification of generation and inference**
- **Ginkgo: Toy Generative Model for Jets**
- **Classification**
- **Maximum likelihood estimate**
- **Visualizations**
- **Preliminary results**
- **Final remarks and future directions**

# Generation and inference unification



Tree joint likelihood:

$$p(x, z|\theta) = \prod_j p(x_j | z_{\text{parent}(x_j)}, \theta) \prod_i p(z_i | z_{\text{parent}(z_i)}, \theta)$$

Aim to invert the generative model:

E.g. estimate the posterior distribution on histories conditioned on the observed particles.

$$p(z|x, \theta) = \frac{p(x|z, \theta) p(z|\theta)}{p(x|\theta)}$$

$$p(x|\theta) = \int dz p(x|z, \theta) p(z, \theta)$$

Sampling with MCMC or probabilistic programming techniques ?

Assuming the toy model, attempt to find the most likely splitting history

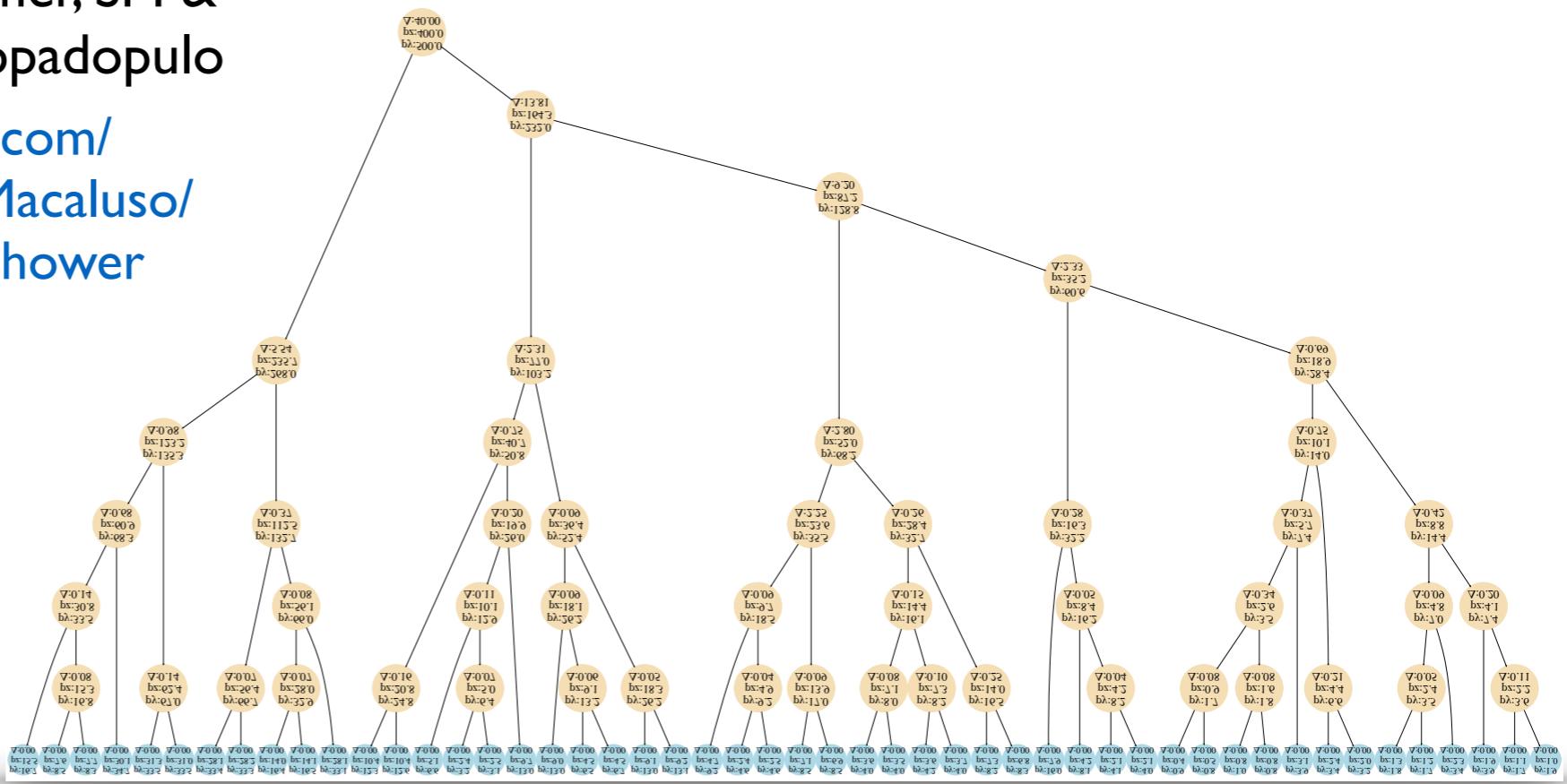
- Reconstructing the splitting history is like inverting the generative model.
- **Hard!** There are a combinatorial number of histories to consider.
- Usually jet reconstruction algorithms don't use the probability model directly.
- We use the likelihood for the history as the optimization **objective**.

# Ginkgo: Toy Generative Model for Jets



Kyle Cranmer, SM &  
Duccio Pappadopulo

[github.com/  
SebastianMacaluso/  
ToyJetsShower](https://github.com/SebastianMacaluso/ToyJetsShower)



## Motivation

- Build a model to aid in ML research for jet physics.

## Generation

- Tractable joint likelihood.
- Captures essential ingredients of parton shower generators in full physics simulations.
- Implements an analogue to a parton shower (no hadronization effects).
- Python implementation with few software dependencies.

## Inference

- E.g. tuning of simulation parameters (PYTHIA TUNES) to optimize a fit to the data.

# Model Specification

- Stand-alone mathematical **specification** of model.
- Document describing model to non-physicists.
- Algorithm described in computer science style.
- Python implementation with few software dependencies.

**Toy Generative Model for Jets**

Kyle Cranmer<sup>1</sup>, Sebastian Macaluso<sup>1</sup> and Duccio Pappadopulo<sup>2</sup>

<sup>1</sup> Center for Cosmology and Particle Physics & Center for Data Science, New York University, USA  
<sup>2</sup> Bloomberg LP, New York, NY 10022, USA.

**1 Introduction**

In this notes, we provide a standalone description of a generative model to aid in machine learning (ML) research for jet physics. The motivation is to build a model that has a tractable likelihood, and is as simple and easy to describe as possible but at the same time captures the essential ingredients of parton shower generators in full physics simulations. The aim is for the model to have a python implementation with few software dependencies.

Parton shower generators are software tools that encode a physics model for the simulation of jets that are produced at colliders, e.g. the Large Hadron Collider at CERN. Jets are a collimated spray of energetic charged and neutral particles. Parton showers generate the particle content of a jet, going through a cascade process, starting from an initial unstable particle. In this description, there is a recursive algorithm that produces binary splittings of an unstable parent particle into two children particles, and a stopping rule. Thus, starting from the initial unstable particle, successive splittings are implemented until all the particles are stable (i.e. the stopping rule is satisfied for each of the final particles). We refer to this final particles as the jet constituents.

As a result of this *showering process*, there could be many latent paths that may lead to a specific jet (i.e. the set of constituents). Thus, it is natural and straightforward to represent a jet and the particular showering path that gave rise to it as a binary tree, where the inner nodes represent each of the unstable particles and the leaves represent the jet constituents.



**Stanford NLP Group**  
@stanfordnlp

Computational Linguistics—Natural Language—Machine Learning—Deep Learning. And misc technology from Silicon Valley. (@chr Manning, @jurafsky & @percyliang)

📍 Stanford, CA, USA ⚡ nlp.stanford.edu

🏢 Joined February 2010

100 Following 77.6K Followers

Followed by Ian Goodfellow, Adji Bousso Dieng, Wojciech Zaremba, and 192 others

[Tweets](#) [Tweets & replies](#) [Media](#) [Likes](#)

Stanford NLP Group Retweeted

Kyle Cranmer @KyleCranmer · 10h  
We developed a standalone description (and pyro implementation) of a generative model to aid in machine learning research for jet physics. NLP researchers can think of ground-truth parse trees with a known language model.  
[github.com/SebastianMacal...](https://github.com/SebastianMacal...)

# Model description

Recursive algorithm to generate a binary tree with jet constituents as the leaves.  
Showering process: binary splittings + stopping rule.

## Features

- Momentum conservation
- Running of the splitting scale

### Algorithm 1: Toy Parton Shower Generator

```
1 function NodeProcessing ( $\vec{p}_p$ ,  $\Delta_p$ ,  $\Delta_{cut}$ ,  $\lambda$ , tree)
  Input : parent momentum  $\vec{p}_p$ , parent splitting scale  $\Delta_p$ , cut-off scale  $\Delta_{cut}$ ,
           rate for the exponential distribution  $\lambda$ , binary tree tree
  2   Add parent node to tree.
  3   if  $\Delta_p > \Delta_{cut}$  then
  4     draw  $\phi$  from uniform distribution in  $\{0, 2\pi\}$ 
  5      $\vec{\Delta}_p = \Delta_p (\sin \phi_p, \cos \phi_p)$ 
  6      $\vec{p}_L = \frac{1}{2}\vec{p}_p - \vec{\Delta}_p$ 
  7      $\vec{p}_R = \frac{1}{2}\vec{p}_p + \vec{\Delta}_p$ 
  8     draw  $\Delta_L$ ,  $\Delta_R$  from the exponential distribution in Eq. 3.
  9     NodeProcessing ( $\vec{p}_L$ ,  $\Delta_L$ ,  $\Delta_{cut}$ ,  $\lambda$ , tree)
 10    NodeProcessing ( $\vec{p}_R$ ,  $\Delta_R$ ,  $\Delta_{cut}$ ,  $\lambda$ , tree)
```

# Generative process

Start with  $\Delta_P = \Delta_{\text{root}}$

Sample  $\phi \sim \text{Uniform}(0, 2\pi)$  to pick a direction for  $\Delta_P$

Heavy resonance X jet

$$\Delta_{\text{root}} = \frac{m_X}{2}$$

## Split parent node

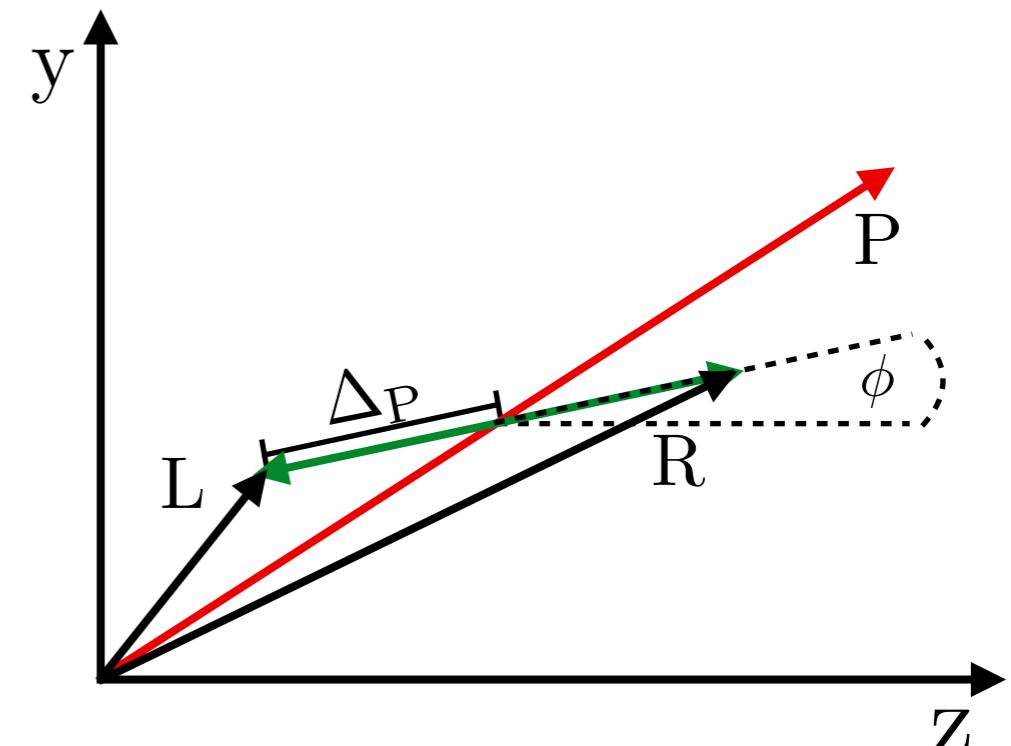
$$\vec{p}_L = \frac{1}{2}\vec{p}_P - \vec{\Delta}_P$$

$$\vec{p}_R = \frac{1}{2}\vec{p}_P + \vec{\Delta}_P$$

Sample independently  $\Delta_L$  and  $\Delta_R$

$$\Delta \sim f(\Delta | \lambda, \Delta_p) = \frac{\lambda}{\Delta_p} e^{-\frac{\lambda}{\Delta_p} \Delta}$$

Recurse over  $(\Delta_L, L)$  and  $(\Delta_R, R)$ .



Three latent variables per particle

$$\vec{p} = (p_y, p_z) \text{ and } \Delta$$

# Augmented data

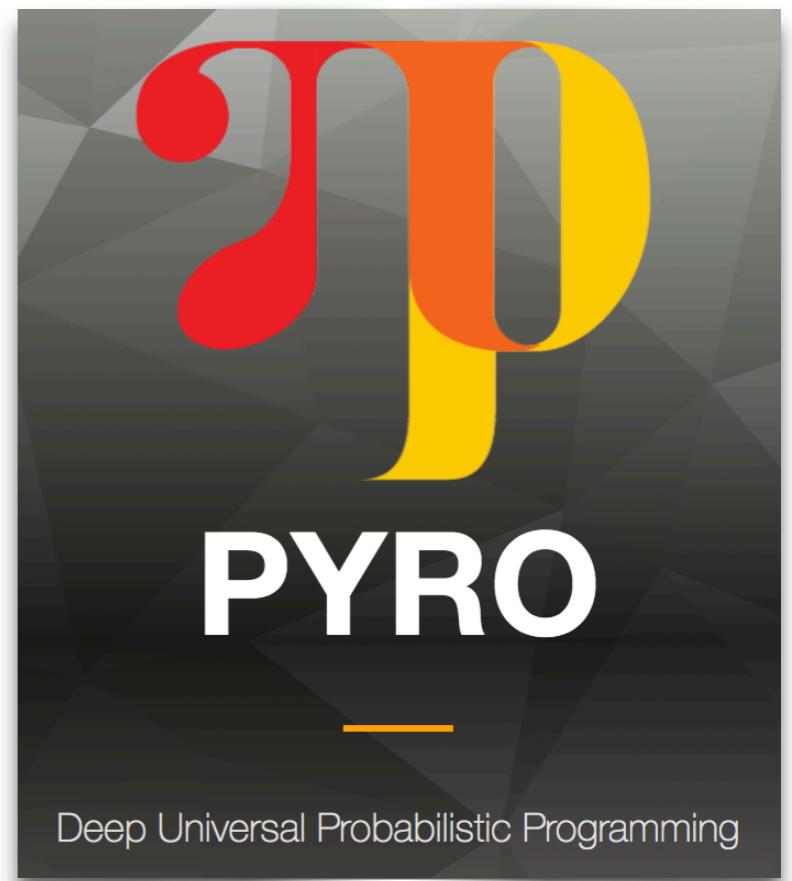
Additional information that characterizes the latent process can be extracted in Ginkgo:

- Joint likelihood
- Joint likelihood ratio
- Joint score

The model keeps track of the augmented data based on a **PYRO** implementation.

New techniques for likelihood-free inference can be applied.

[J. Brehmer, G. Louppe, J. Pavez & K. Cranmer 2018 ]

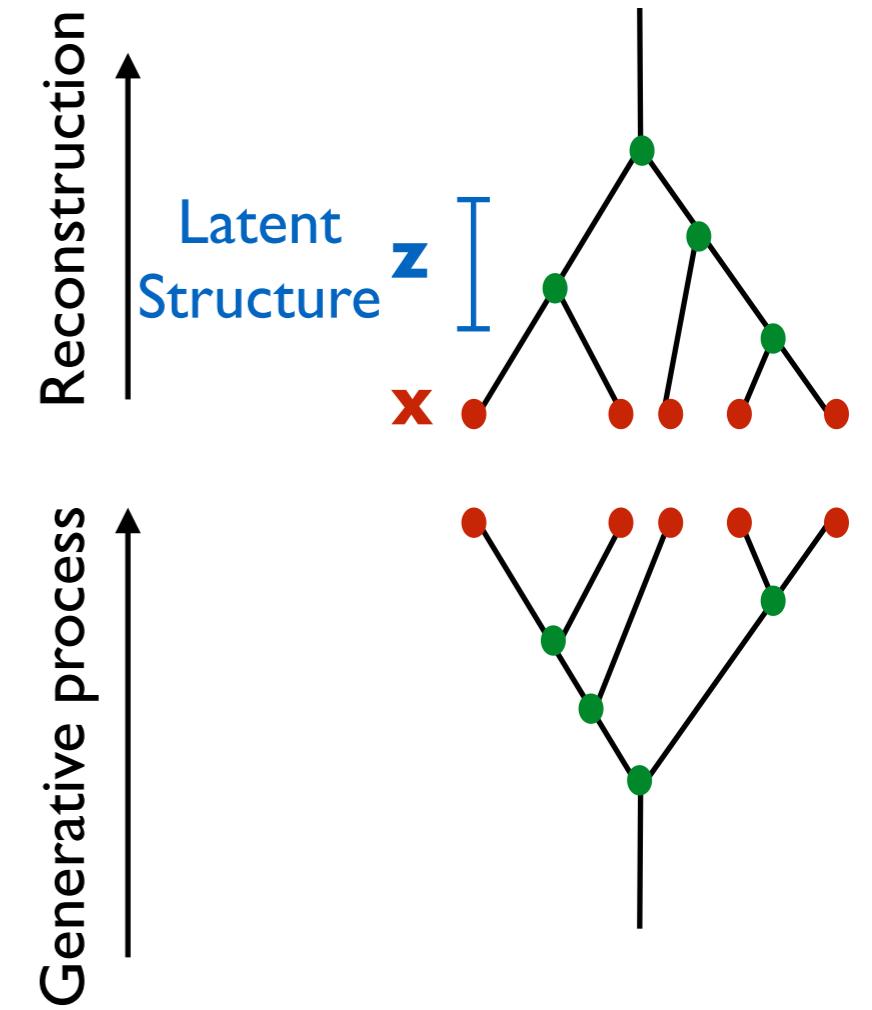


# Problem settings

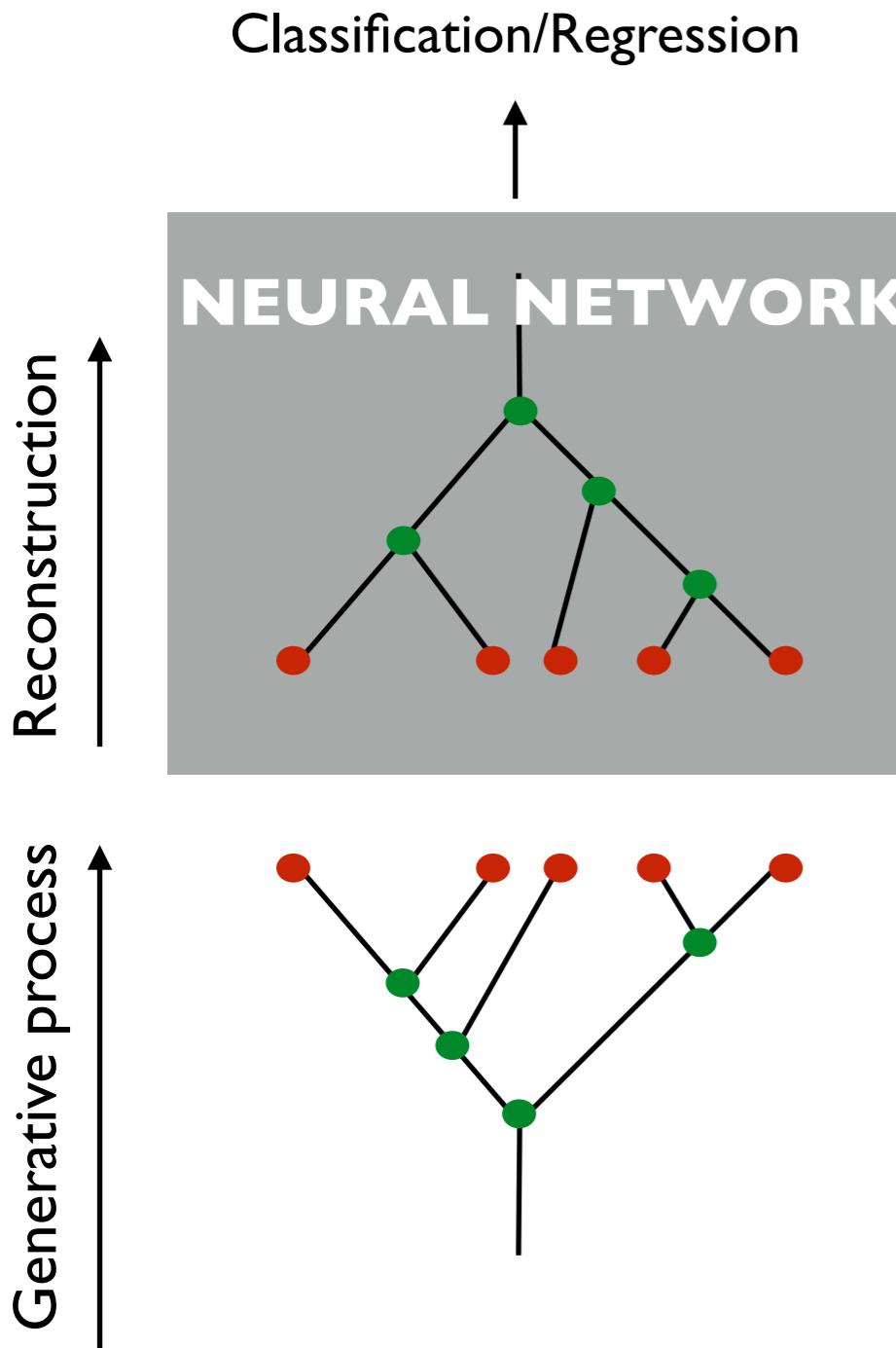
**Classification**

**Maximum likelihood estimate**

**Posterior distribution on histories**



# Neural Networks that conserve the tree structure

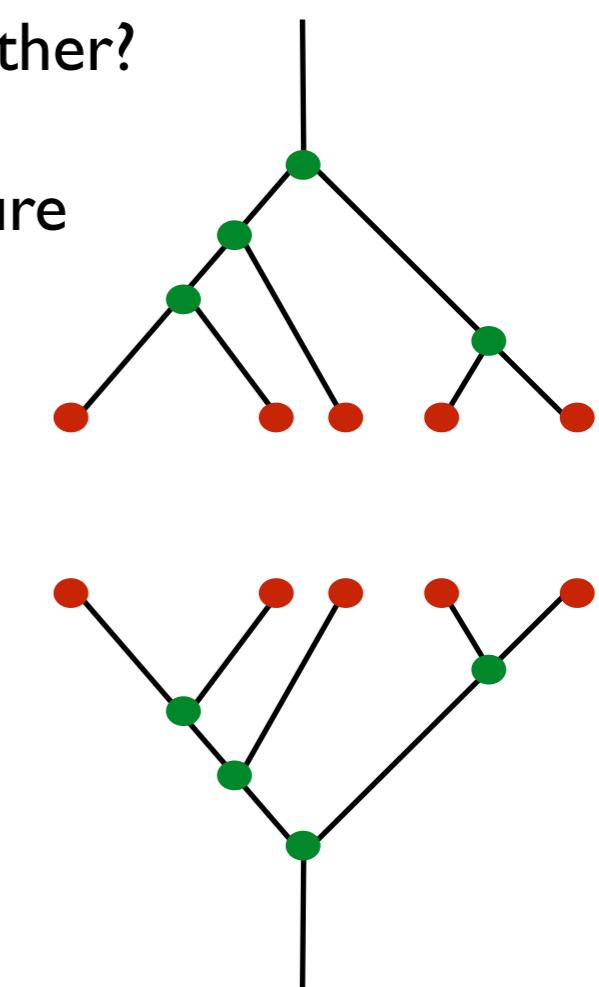


Classification/Regression

Can we solve both tasks together?

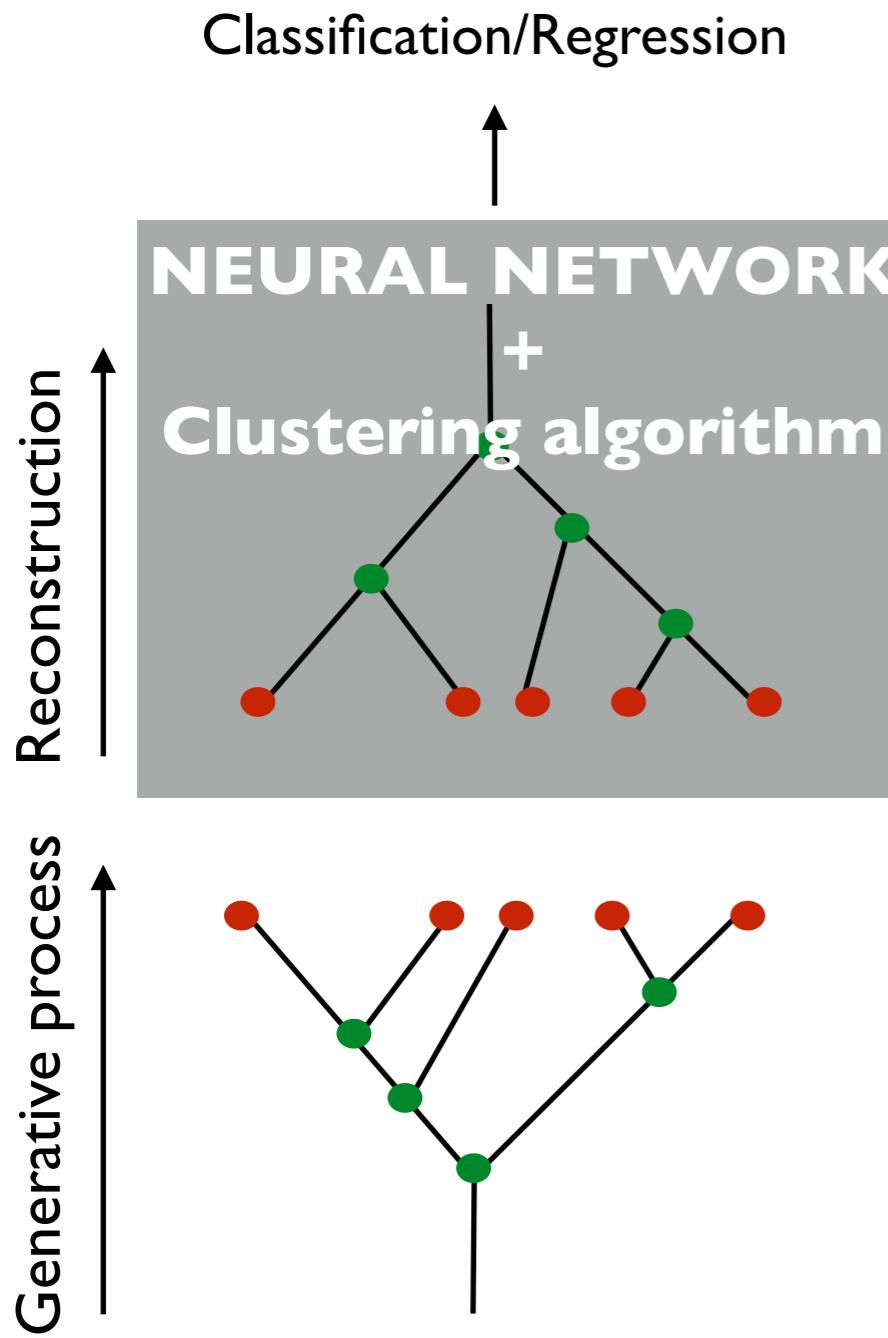
Find the right latent structure  
+  
classification/regression

The diagram illustrates the generative process and reconstruction tasks. On the left, a large grey arrow points upwards, labeled "Generative process" vertically along its left side. On the right, another grey arrow points upwards, labeled "Reconstruction" vertically along its left side. Between these arrows, the text "Can we solve both tasks together?" is centered. Below it, the text "Find the right latent structure + classification/regression" is also centered. This part of the diagram represents the dual nature of the learning task.

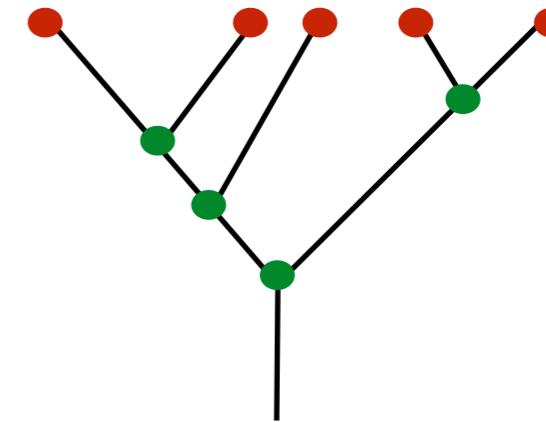
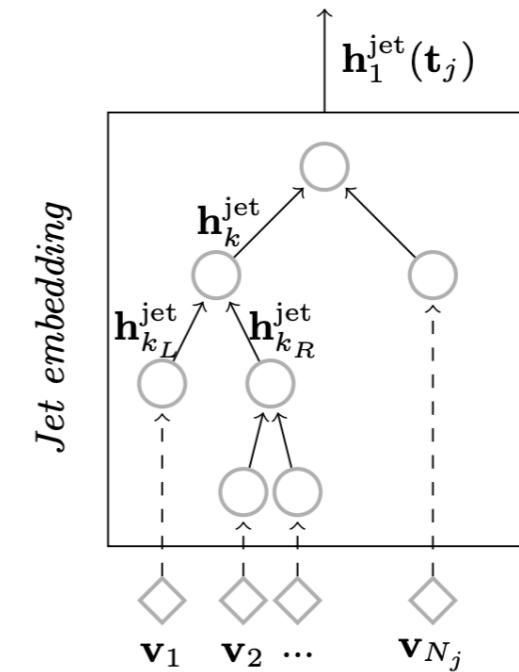


# Neural Networks that conserve the tree structure

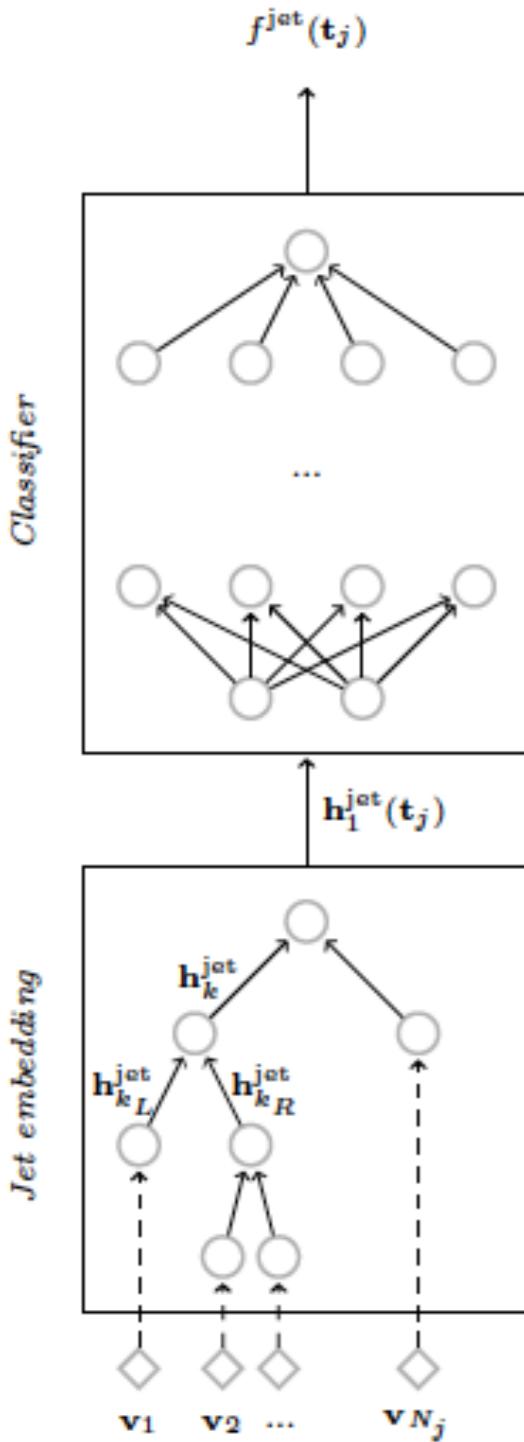
RecNN [Louppe, Cho, Becot & Cranmer '17]  
TreeNiN [Macaluso & Cranmer '19]



Classification/Regression



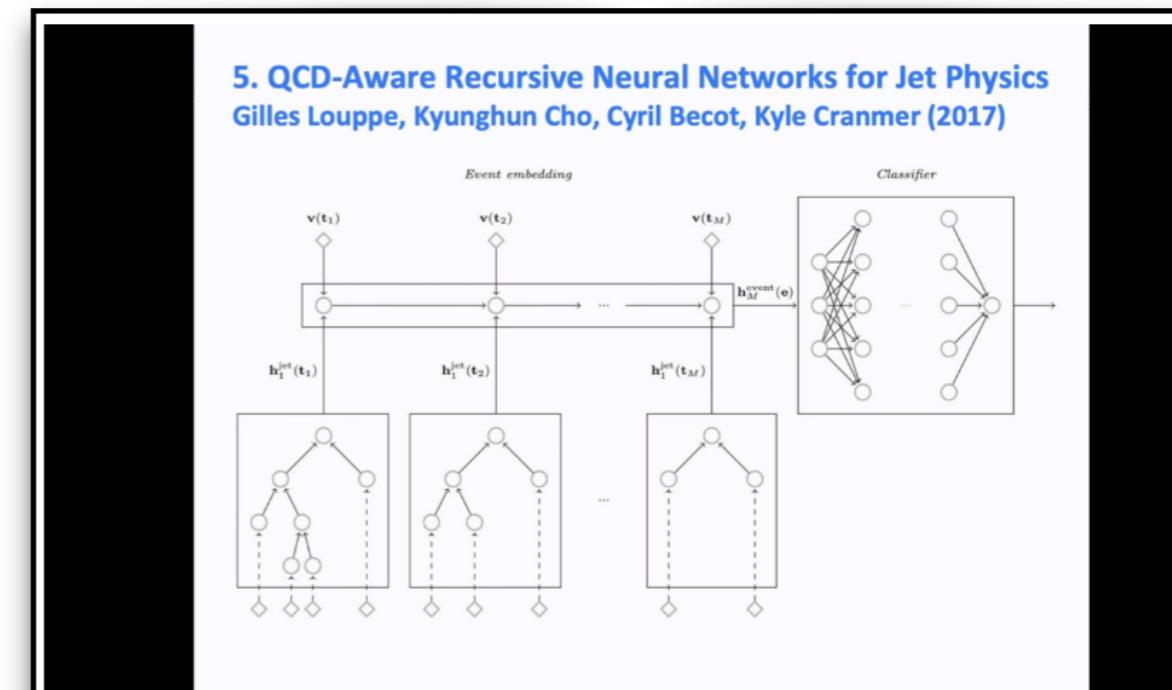
# Recursive Neural Networks for Jet Physics



$$h_k^{\text{jet}} = \begin{cases} u_k & \text{if } k \text{ is a leaf} \\ \sigma \left( W_h \begin{bmatrix} h_{k_L}^{\text{jet}} \\ h_{k_R}^{\text{jet}} \\ u_k \end{bmatrix} + b_h \right) & \text{otherwise} \end{cases}$$

$$u_k = \sigma (W_u g(\mathbf{o}_k) + b_u)$$

- RecNN much fewer trainable parameters:  
 $\sim 10000$  vs  $\sim 1 M$
- Low level input features:  
 $|\vec{p}|, \eta, \phi, E, \frac{E}{E_{\text{jet}}}, p_T, \theta$

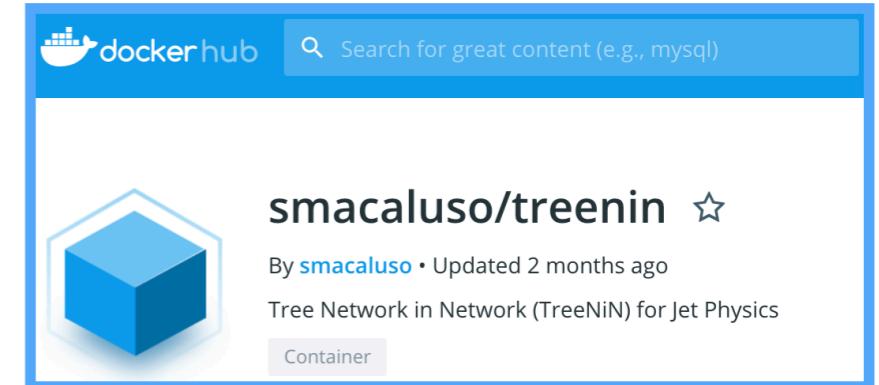


Stanford CS224N: NLP with Deep Learning  
 Winter 2019 | Lecture 18 – Constituency  
 Parsing, TreeRNNs

# Tree Network in Network (TreeNiN) for Jet Physics

[Macaluso & Cranmer '19]

- GPU implementation of a (RecNN + 2 NiN layers).
- Split weights between leaves, inners nodes and main subtrees (both for the initial embedding and the NiN layers).



<https://github.com/diana-hep/TreeNiN>

## Network in Network (NiN) Mechanism

- Extra layers within each node + non-linear activation function

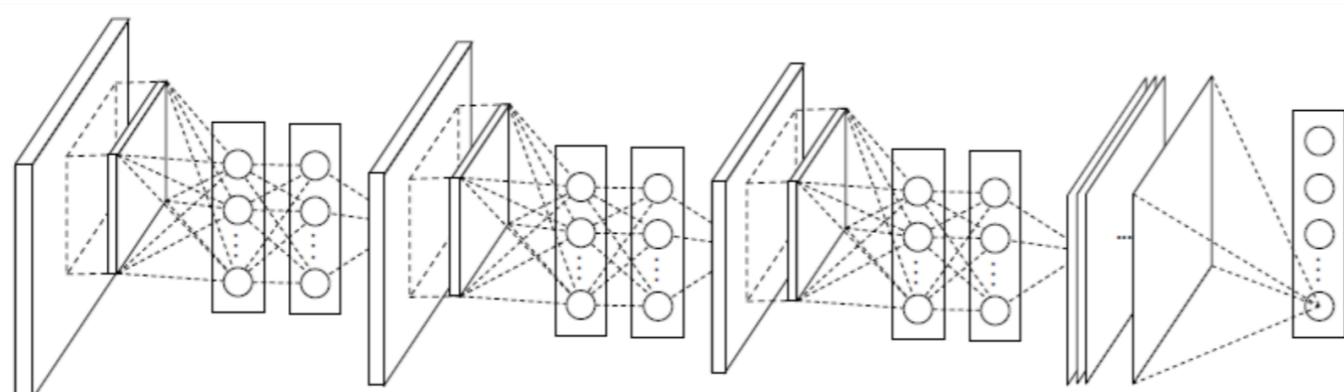
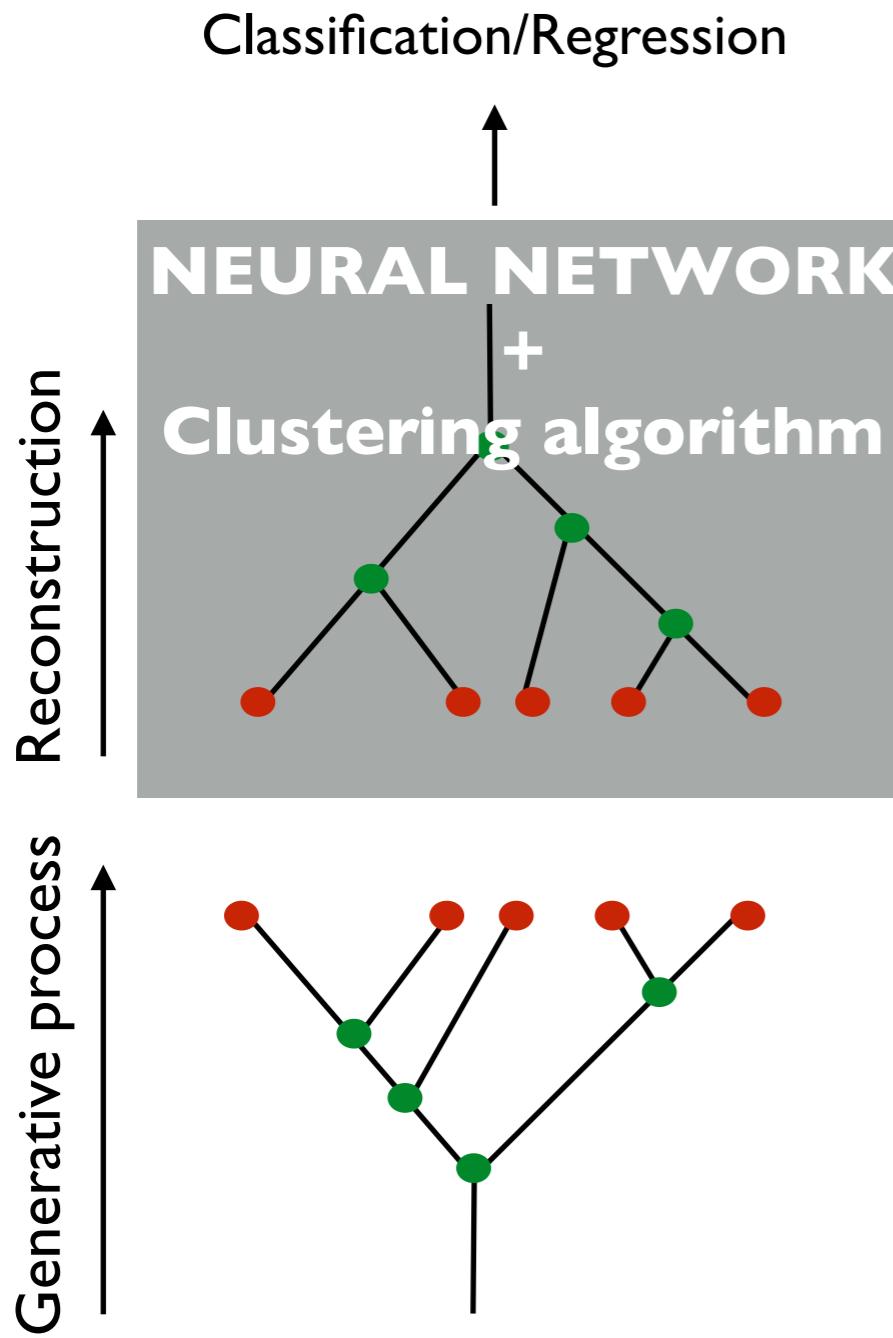


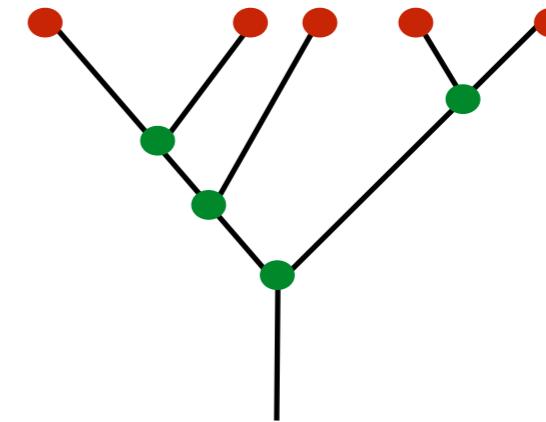
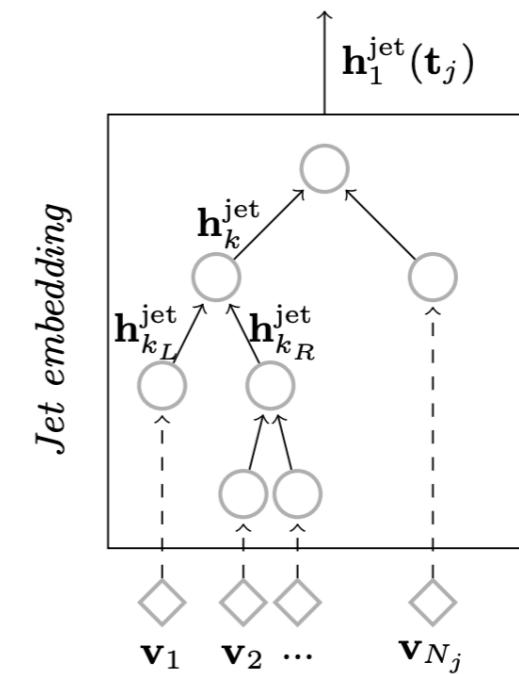
Fig. from arXiv:1312.4400

# Neural Networks that conserve the tree structure

RecNN [Louppe, Cho, Becot & Cranmer '17]  
TreeNiN [Macaluso & Cranmer '19]



Classification/Regression



# Generalized kt clustering algorithms

- “Heuristic” domain-specific clustering algorithms that do not use the generative model.
- Idea: sequentially cluster jet constituents.
- Permutation invariance: add 4-momenta of each pair that is clustered.
- Merge closest pair based on the distance measure:

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

$$\Delta R_{ij}^2 = (y_i - y_j)^2 + (\phi_i - \phi_j)^2$$

$\alpha = \{1, 0, -1\}$  specifies the the kt, Cambridge/Aachen and anti-kt algorithms respectively.

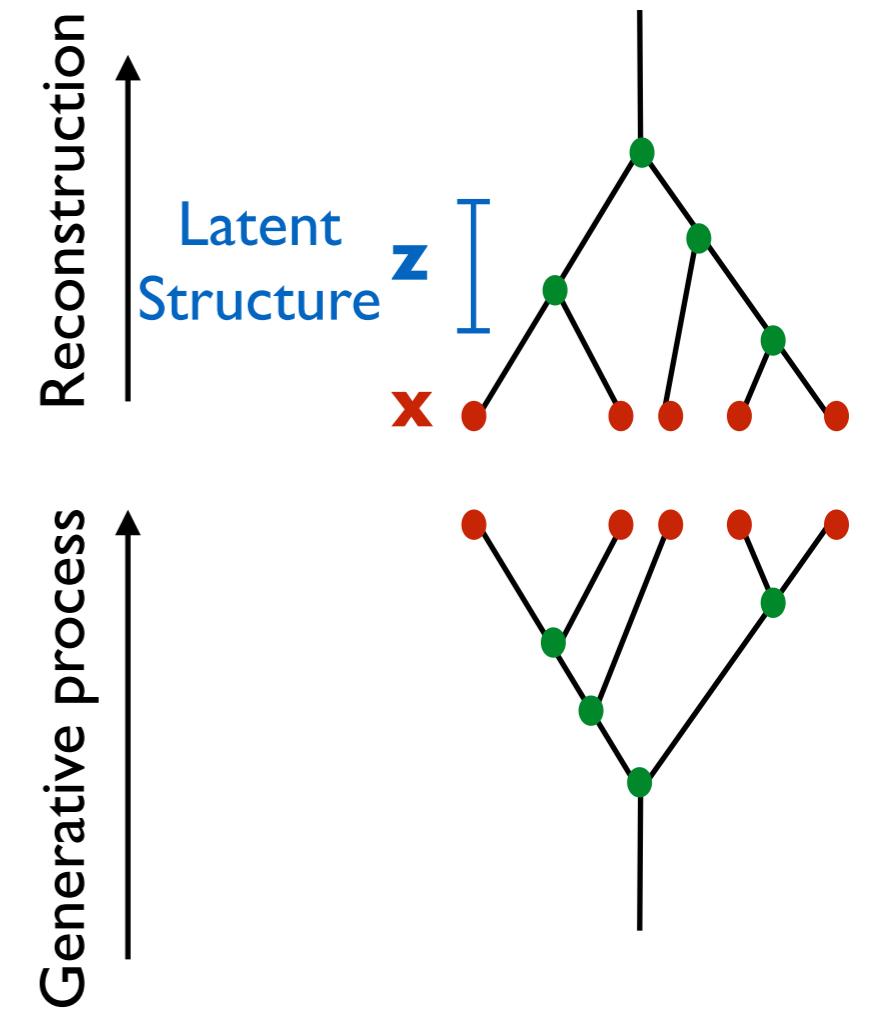
The screenshot shows the INSPIRE HEP search results page. The top navigation bar includes the INSPIRE logo, a HEP icon, and a HEP NAMES link. Below the navigation is a search bar with the query "find eprint 0802.1189" and a "find j "Phys.Rev.Lett.,105\*\*" link. To the right are sorting options ("Sort by: latest first, desc., - or rank by") and display settings ("Display results: 25 results, single list"). The main content area displays a single result under the heading "HEP". The result title is "1 records found". The entry is "1. The anti- $k_t$  jet clustering algorithm" by Matteo Cacciari, Gavin P. Salam (Paris, LPTHE), Gregory Soyez (LPTHE-07-03). It includes the DOI (10.1088/1126-6708/2008/04/063), arXiv link (arXiv:0802.1189 [hep-ph]), PDF link, References, BibTeX, LaTeX(US), LaTeX(EU), Harvmac, and ADS Abstract Service links. A red circle highlights the "Detailed record - Cited by 6197 records" link at the bottom right of the result card.

# Problem settings

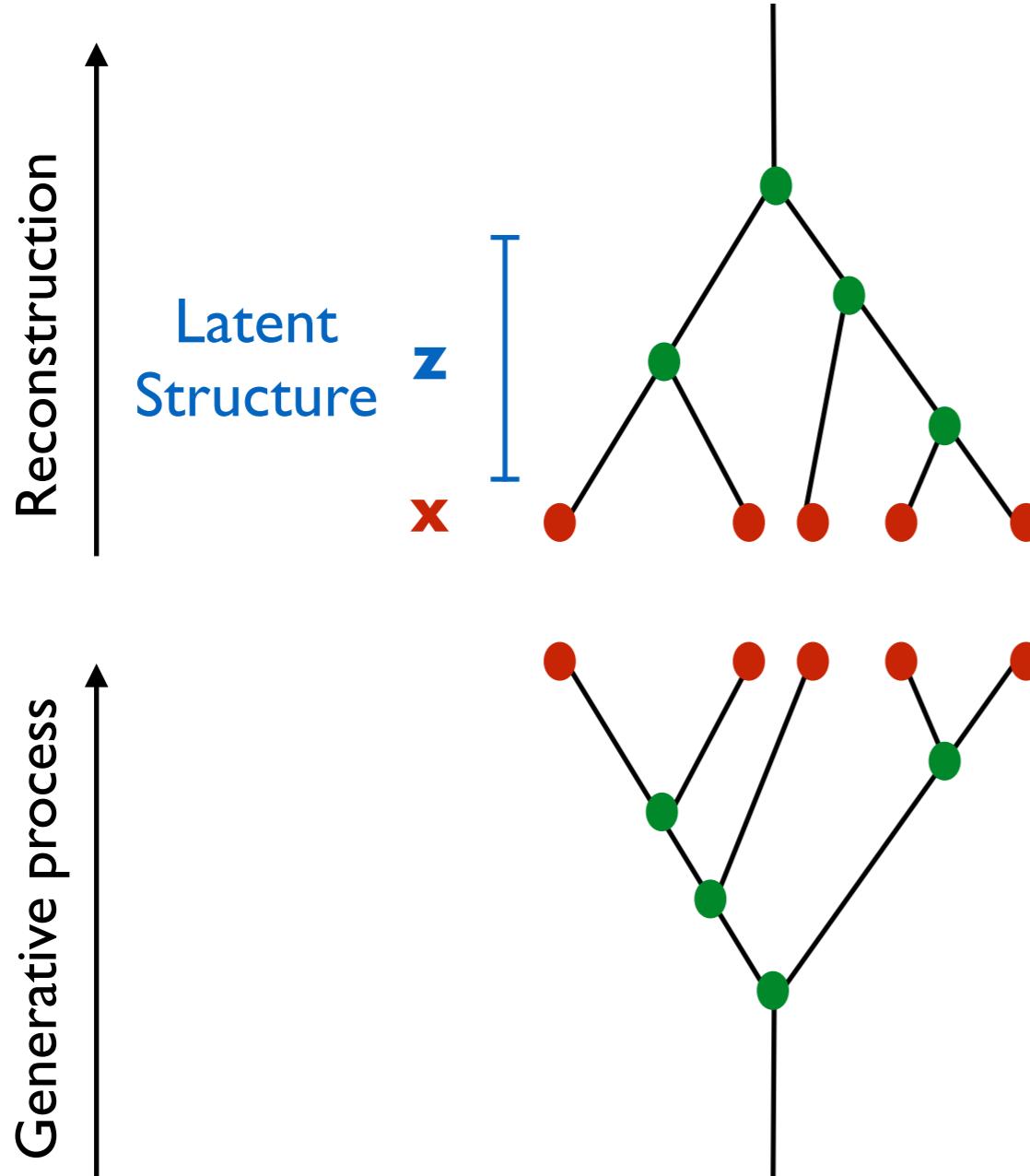
**Classification**

**Maximum likelihood estimate**

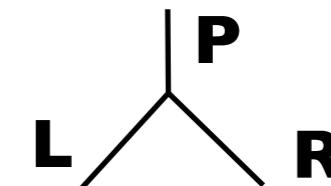
**Posterior distribution on histories**



# Jet clustering



Can we cluster jets based on the likelihood of each splitting?



Joint likelihood:

$$p(x, z | \theta) = \prod_i p(L_i, R_i | P_i, \theta)$$

Split likelihood:

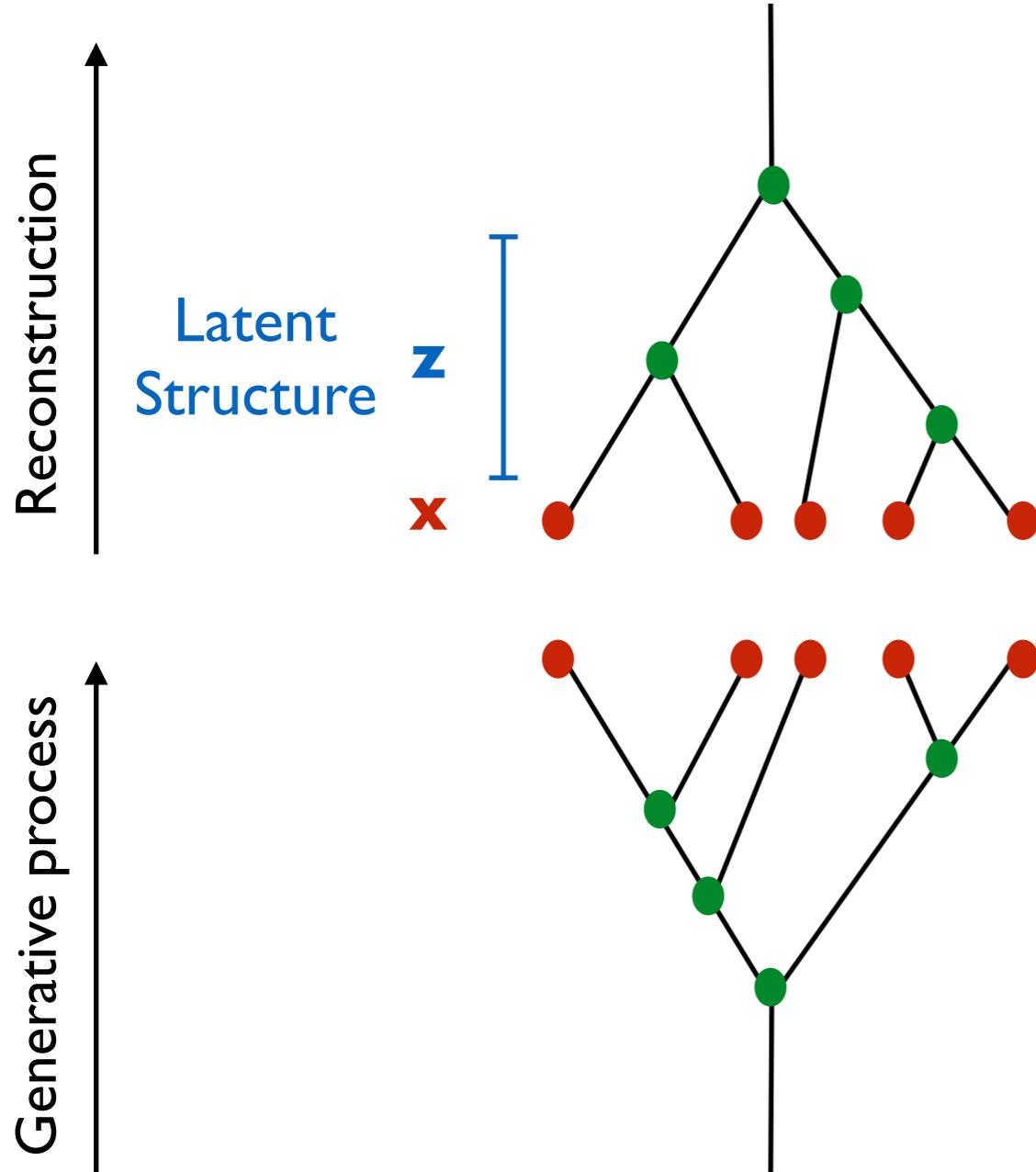
$$p(L_i, R_i | P_i) \equiv p(\vec{p}_L, \vec{p}_R | \vec{p}_P)$$

---

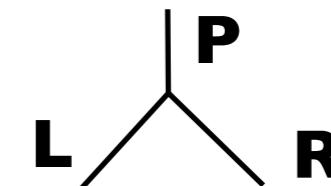
$k_t$  distance measure

$$d(L_i, R_i) \equiv d(\vec{p}_L, \vec{p}_R)$$

# Jet clustering



Can we cluster jets based on the likelihood of each splitting?



Joint likelihood:

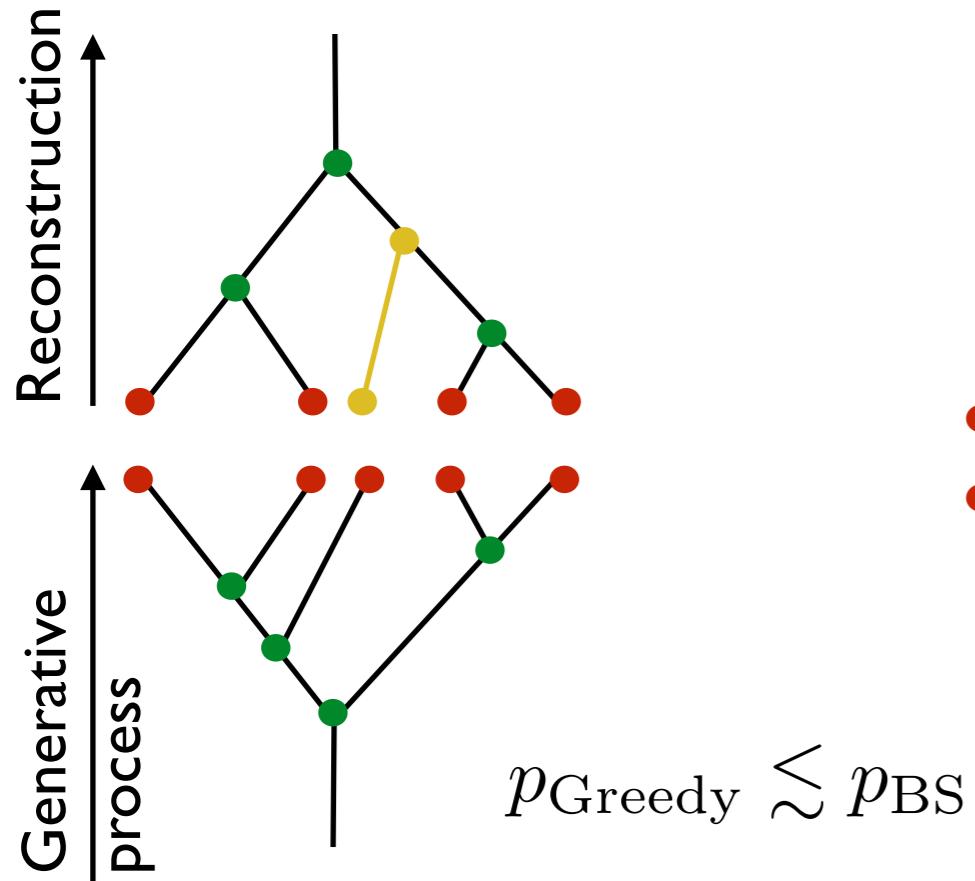
$$p(x, z | \theta) = \prod_i p(L_i, R_i | P_i, \theta)$$

- Hard to do with parton shower generators in full physics simulations, e.g. PYTHIA
- To prototype we built our own toy model!

# Likelihood-based jet clustering

## Greedy

Joint likelihood from locally maximizing the likelihood at each step.



## Beam Search

Maximize the likelihood of multiple steps before choosing the latent path.  
Maybe we could fix mistakes?

Joint likelihood:

$$p(x, z|\theta) = \prod_i p(L_i, R_i | P_i, \theta)$$

Goal: find the latent structure that maximizes the jet likelihood (MLE).

Given an algorithm,  $z$  is fixed.

$\hat{z}_{\text{Greedy}}$

$\hat{z}_{\text{BS}}$

$$\hat{z}_{\text{MLE}} = \operatorname{argmax}_z p(x|z)$$

Viterbi algorithm

Computationally “infeasible” for jets with  $\sim 100$  constituents

# Greedy algorithms

Choose the optimal decision locally at each step.

## Jets properties

- Permutation invariance
- Distance measure

## Generalized kt clustering algorithms

- Locally minimize the distance measure  $d_{ij}$
- An analogue of the generalized  $k_t$  clustering algorithms can be defined for Ginkgo jets.

## Greedy likelihood-based algorithm

- Choose the nodes pairing that locally maximizes the likelihood at each step given by

$$p(\Delta_L, \Delta_R | \Delta_p)$$

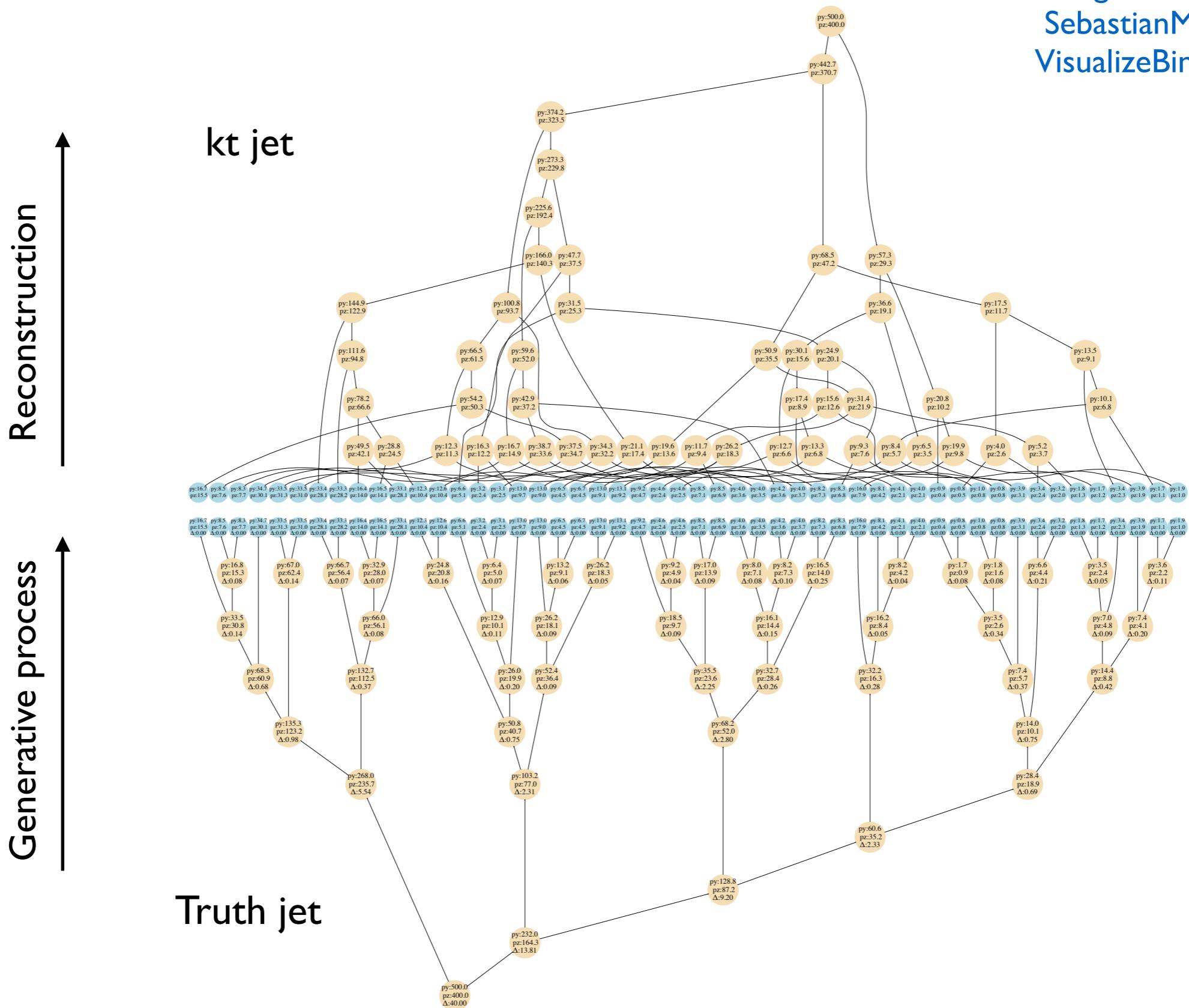
Our implementation improves the  $N^3$  brute force time complexity to  $N^2$ .

Approach based on nearest neighbors introduced in FastJet  
[hep-ph/0512210, G. Salam and M. Cacciari LPTHE-06-02].

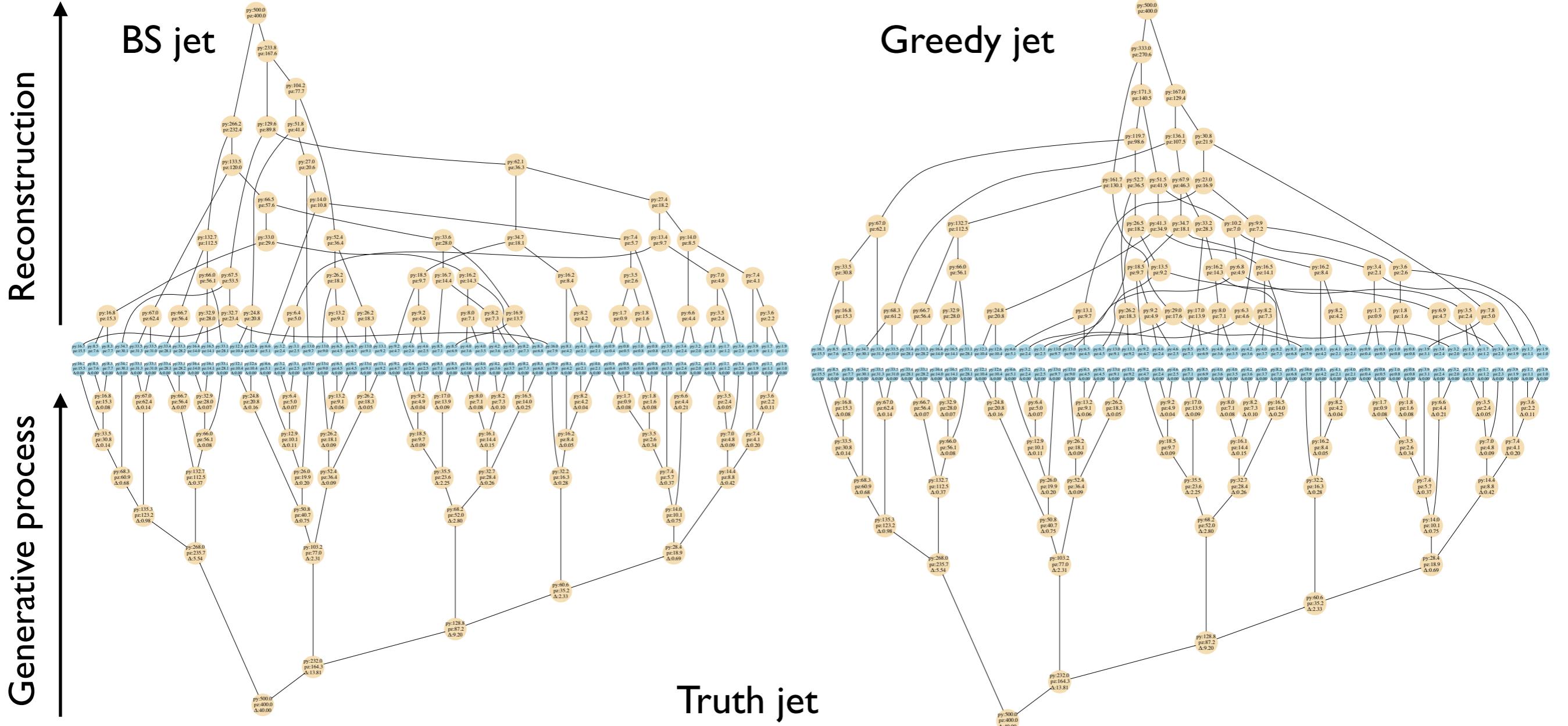
# 1-D Tree only visualizations

Kyle Cranmer, SM &  
Duccio Pappadopulo

[github.com/  
SebastianMacaluso/  
VisualizeBinaryTrees](https://github.com/SebastianMacaluso/VisualizeBinaryTrees)



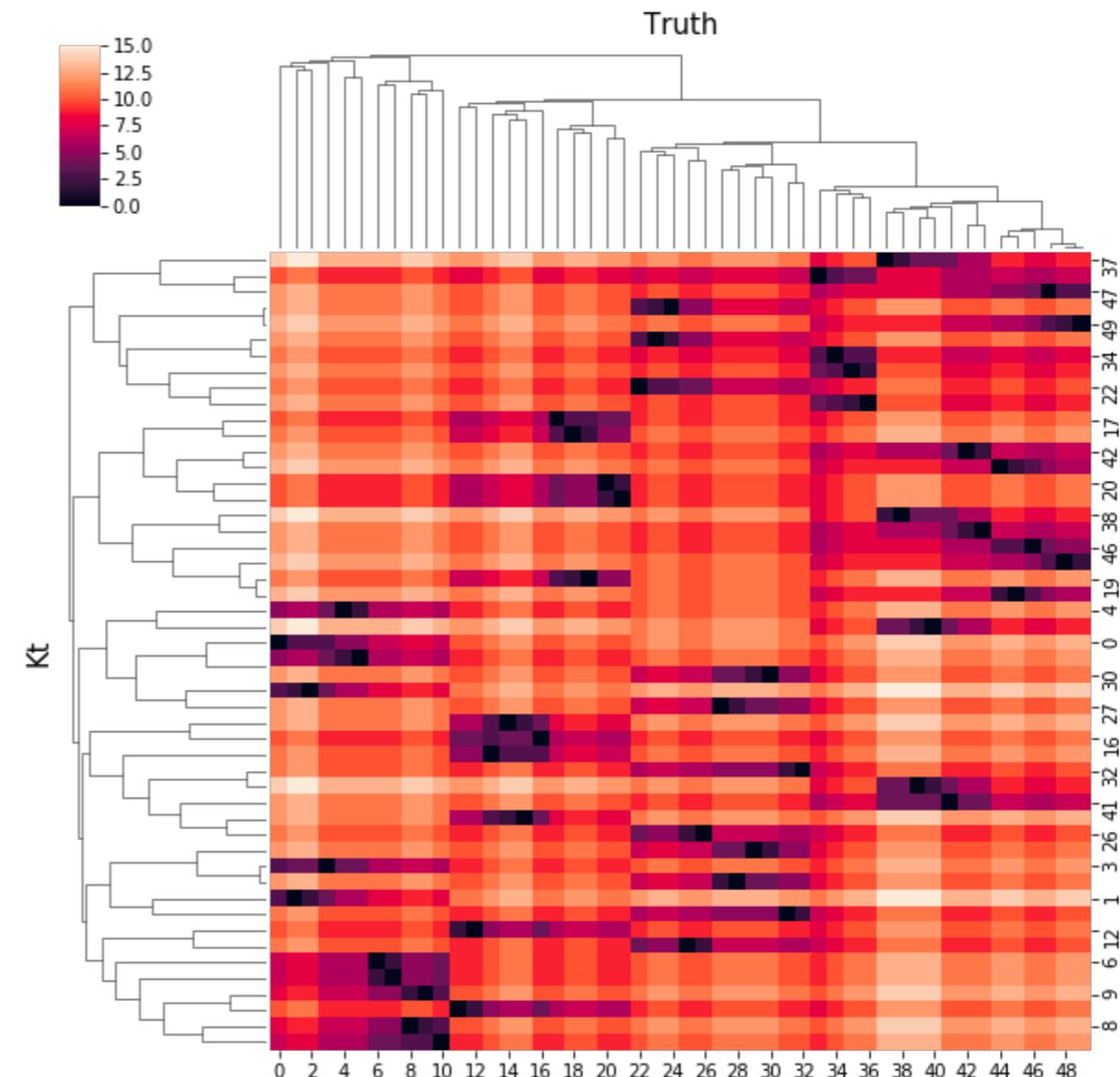
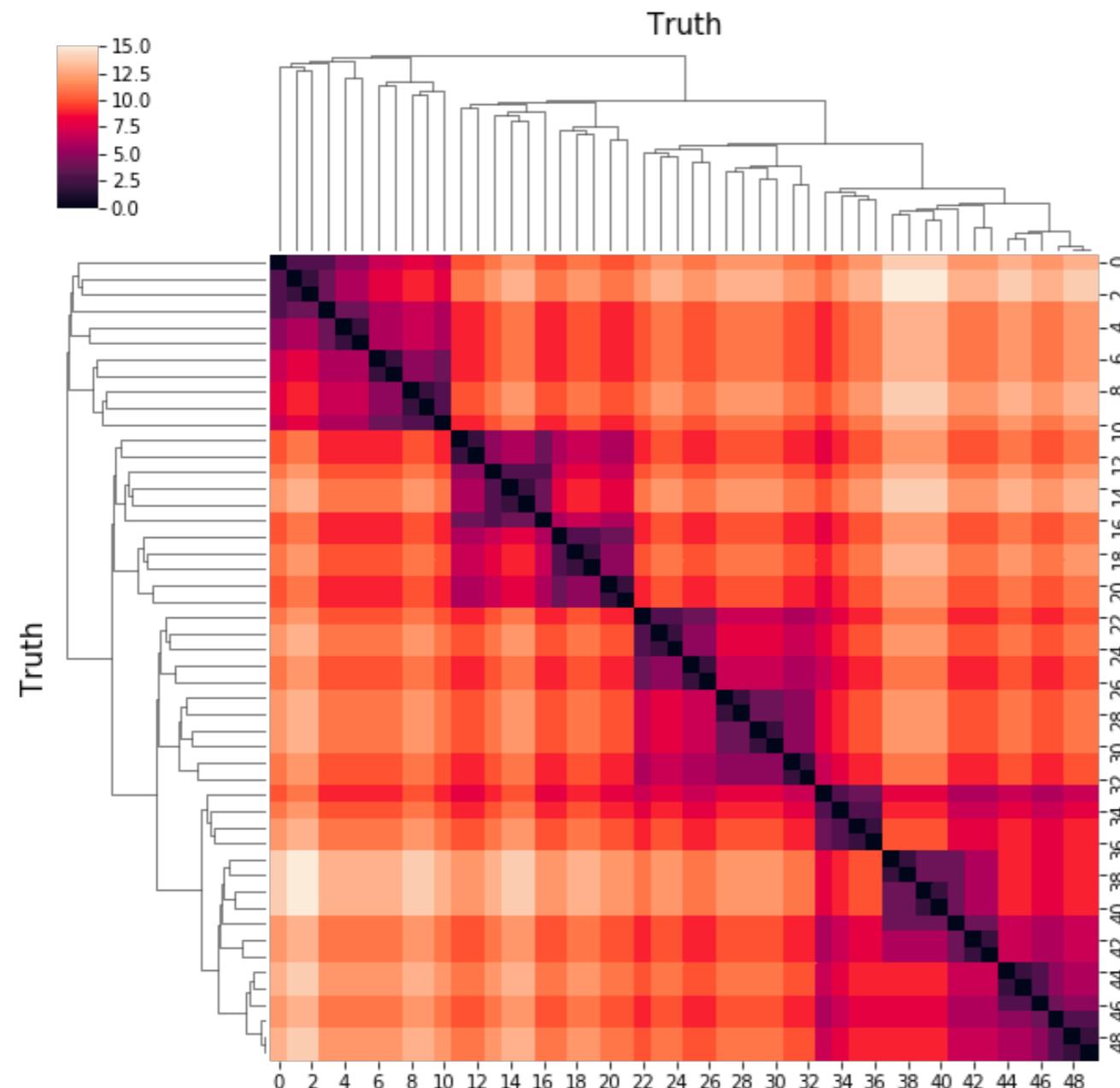
# 1-D Tree only visualizations



# Heat cluster maps visualizations

Kyle Cranmer, SM &  
Duccio Pappadopulo

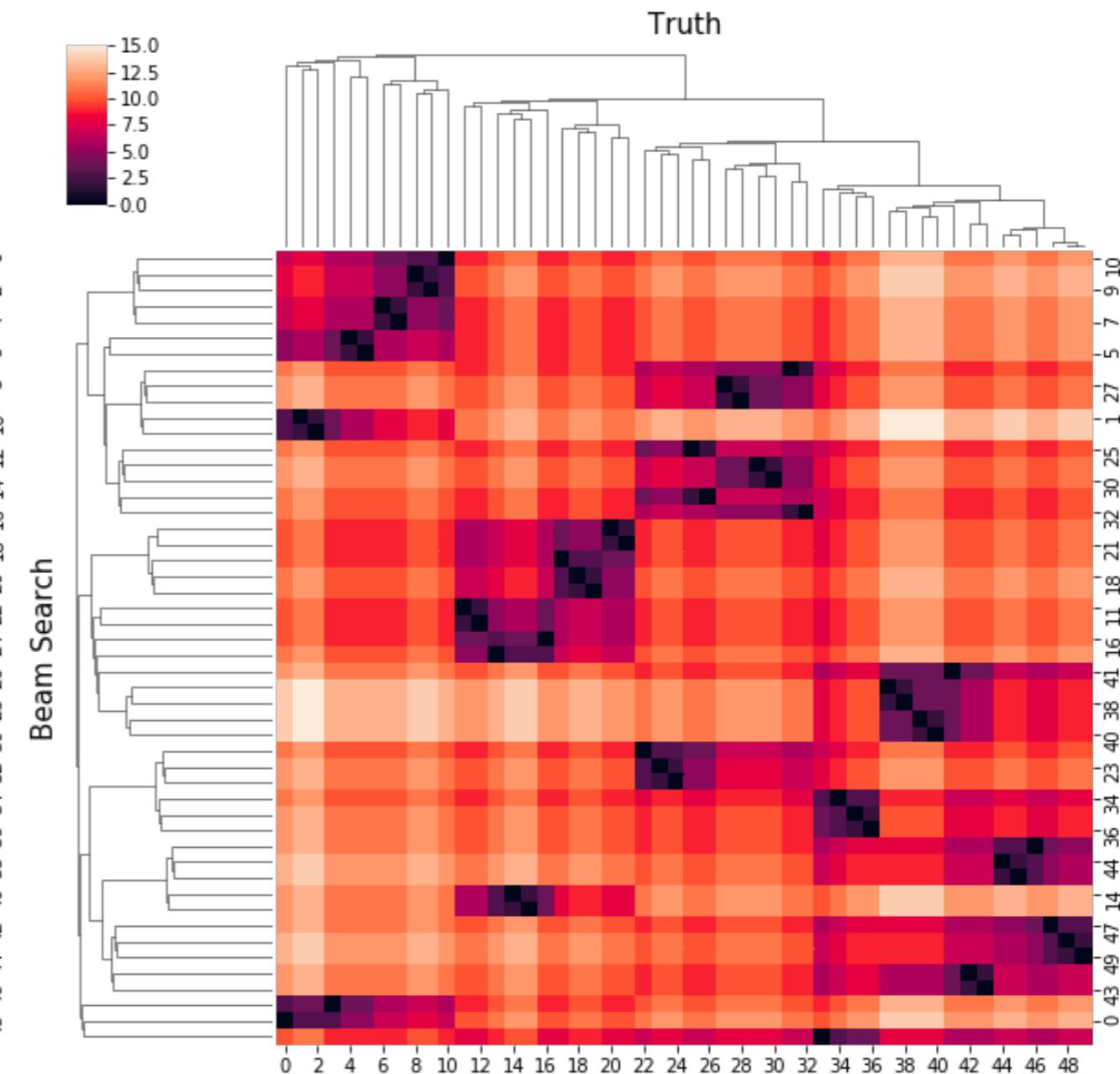
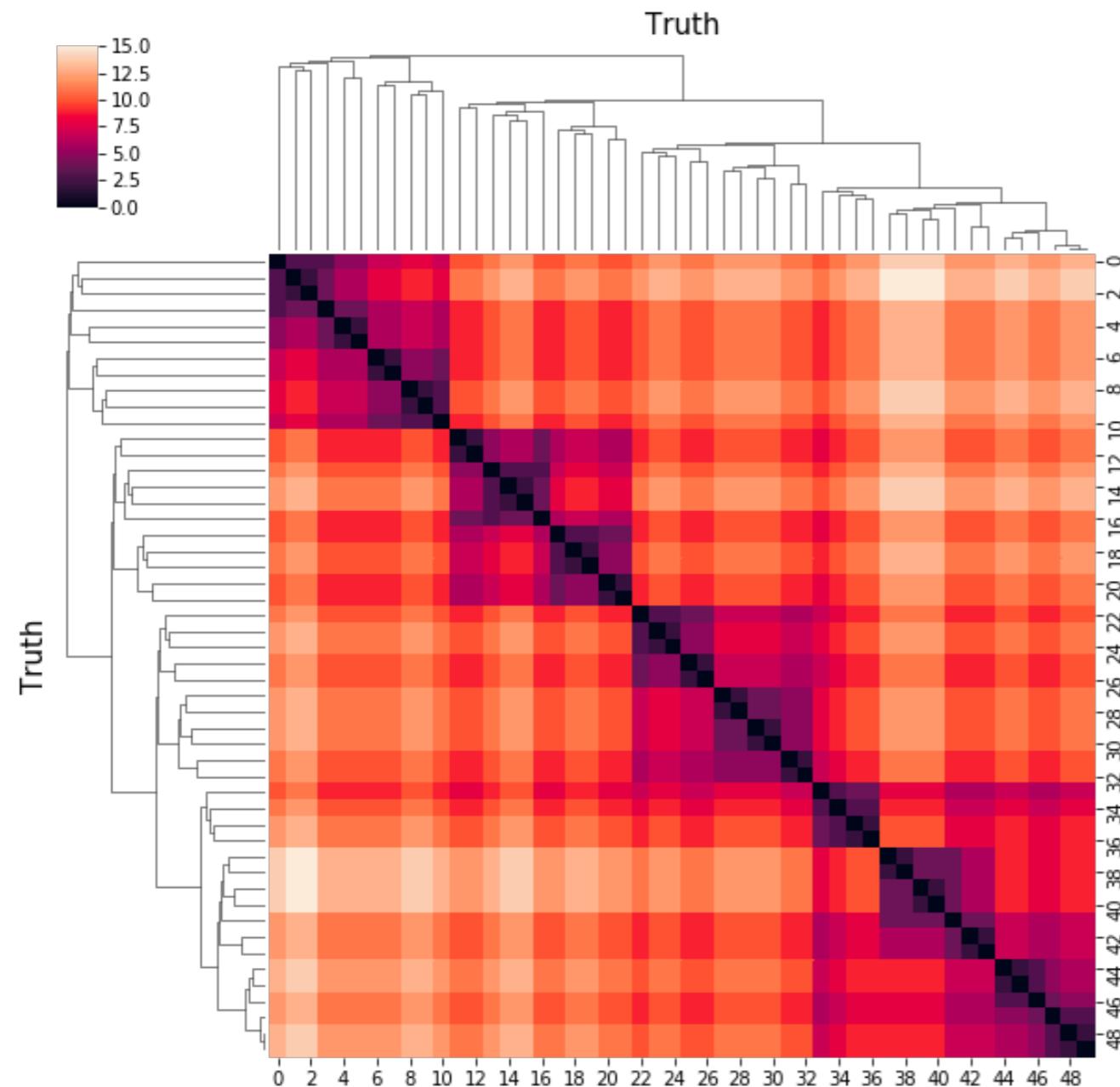
[github.com/SebastianMacaluso/  
VisualizeBinaryTrees](https://github.com/SebastianMacaluso/VisualizeBinaryTrees)



# Heat cluster maps visualizations

Kyle Cranmer, SM &  
Duccio Pappadopulo

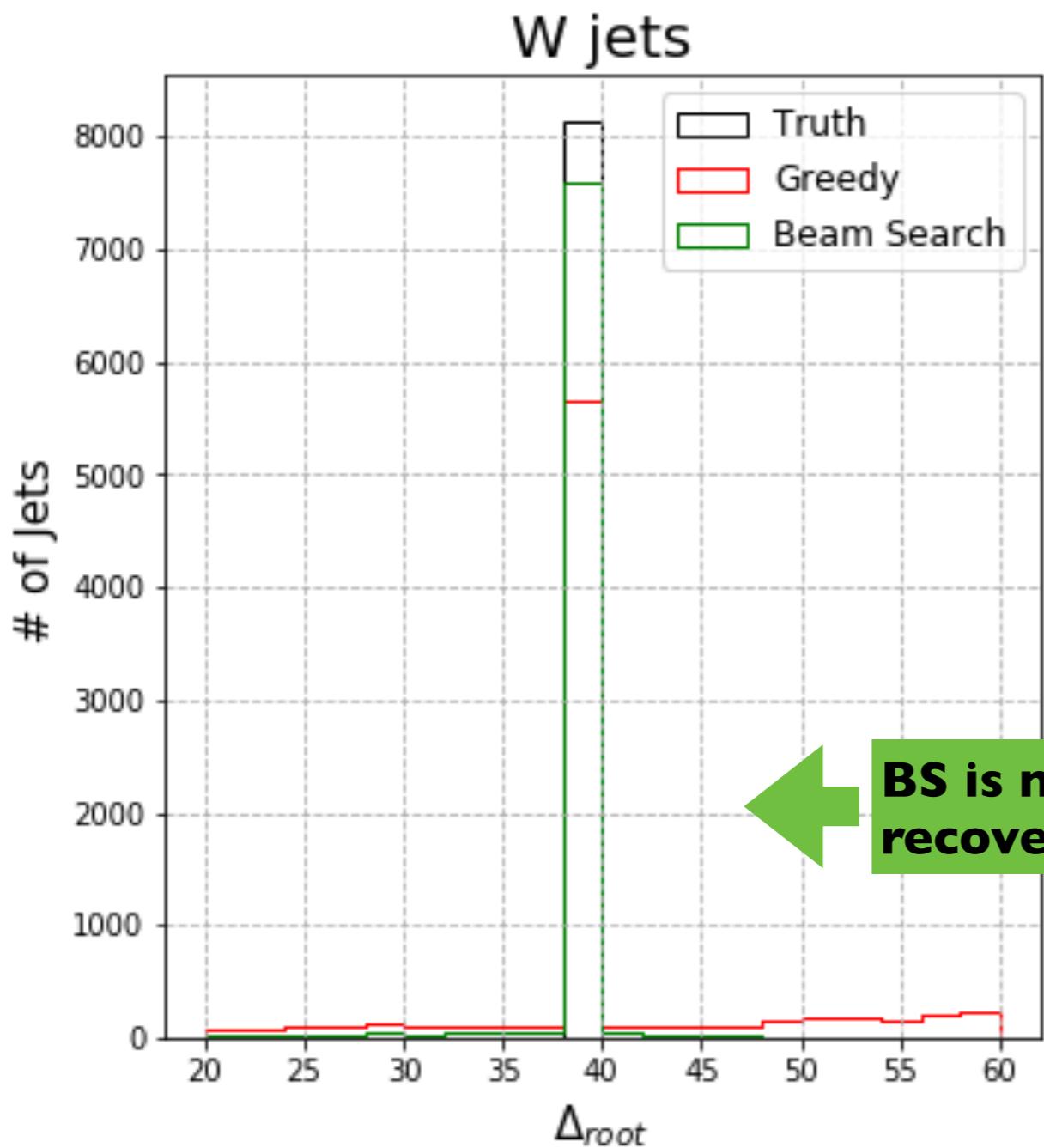
[github.com/SebastianMacaluso/  
VisualizeBinaryTrees](https://github.com/SebastianMacaluso/VisualizeBinaryTrees)



# Reconstruction of $\Delta_{\text{root}}$

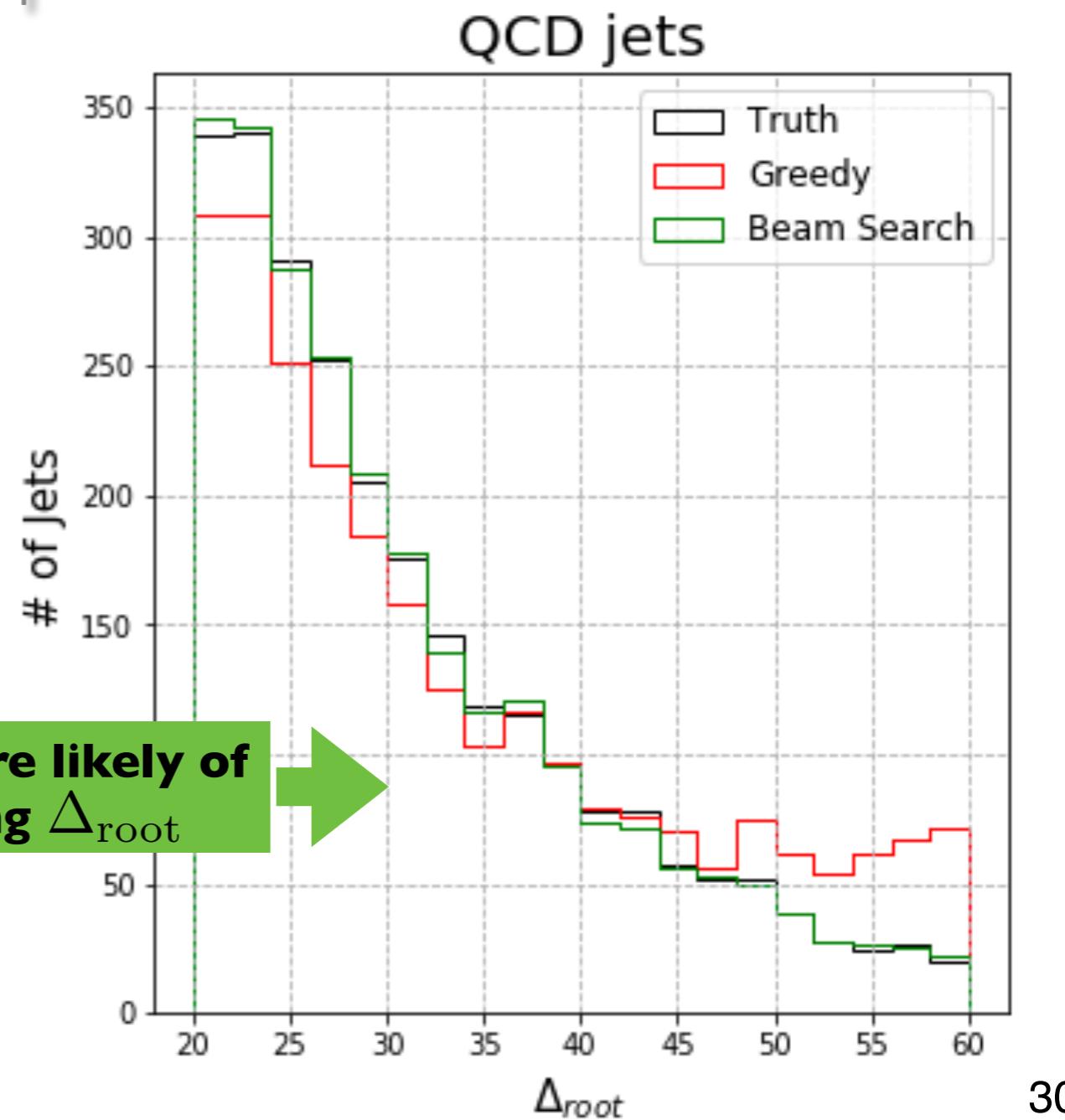
Ginkgo W jets

$$\Delta_{\text{root}} = \frac{m_W}{2}$$

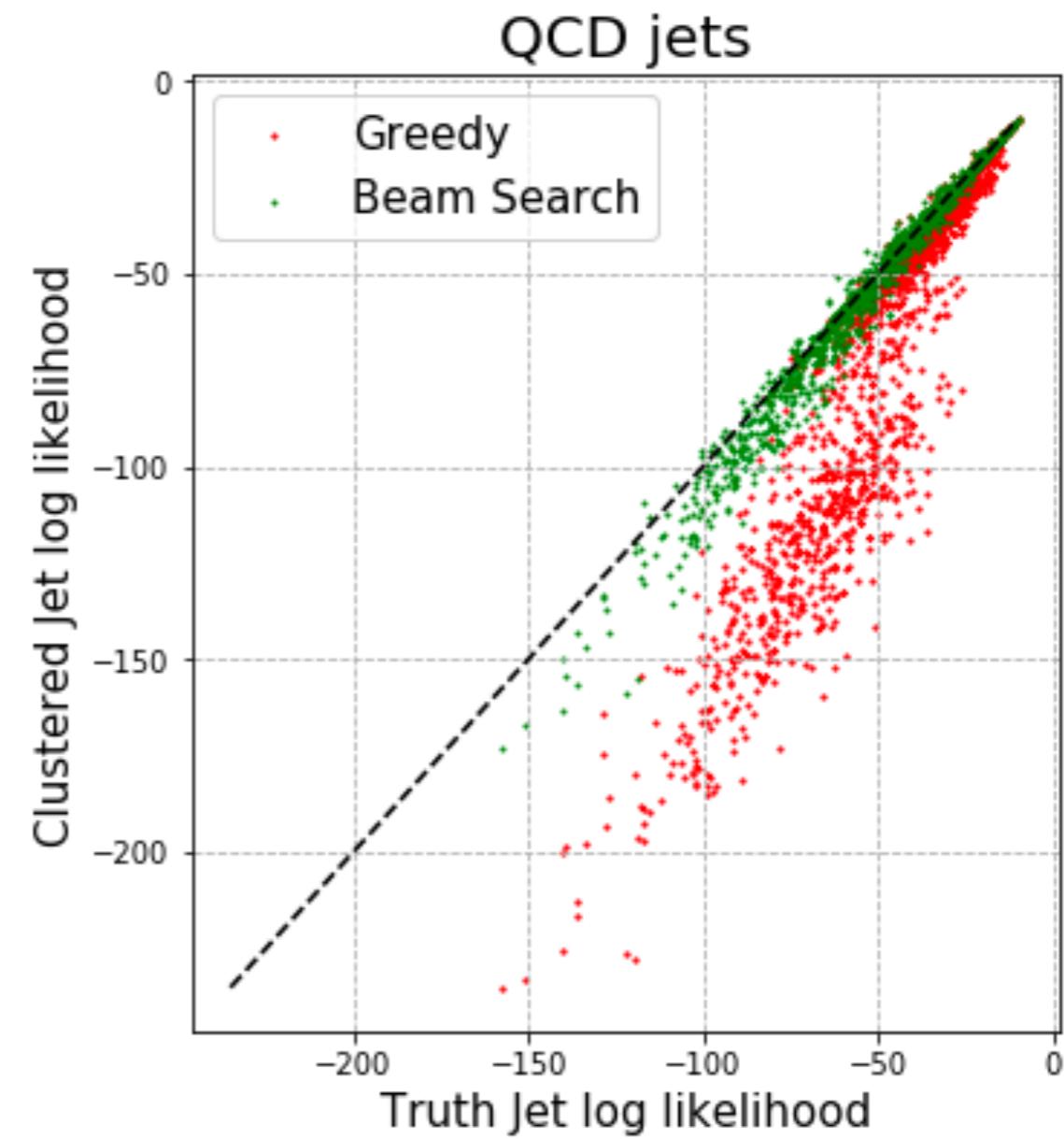
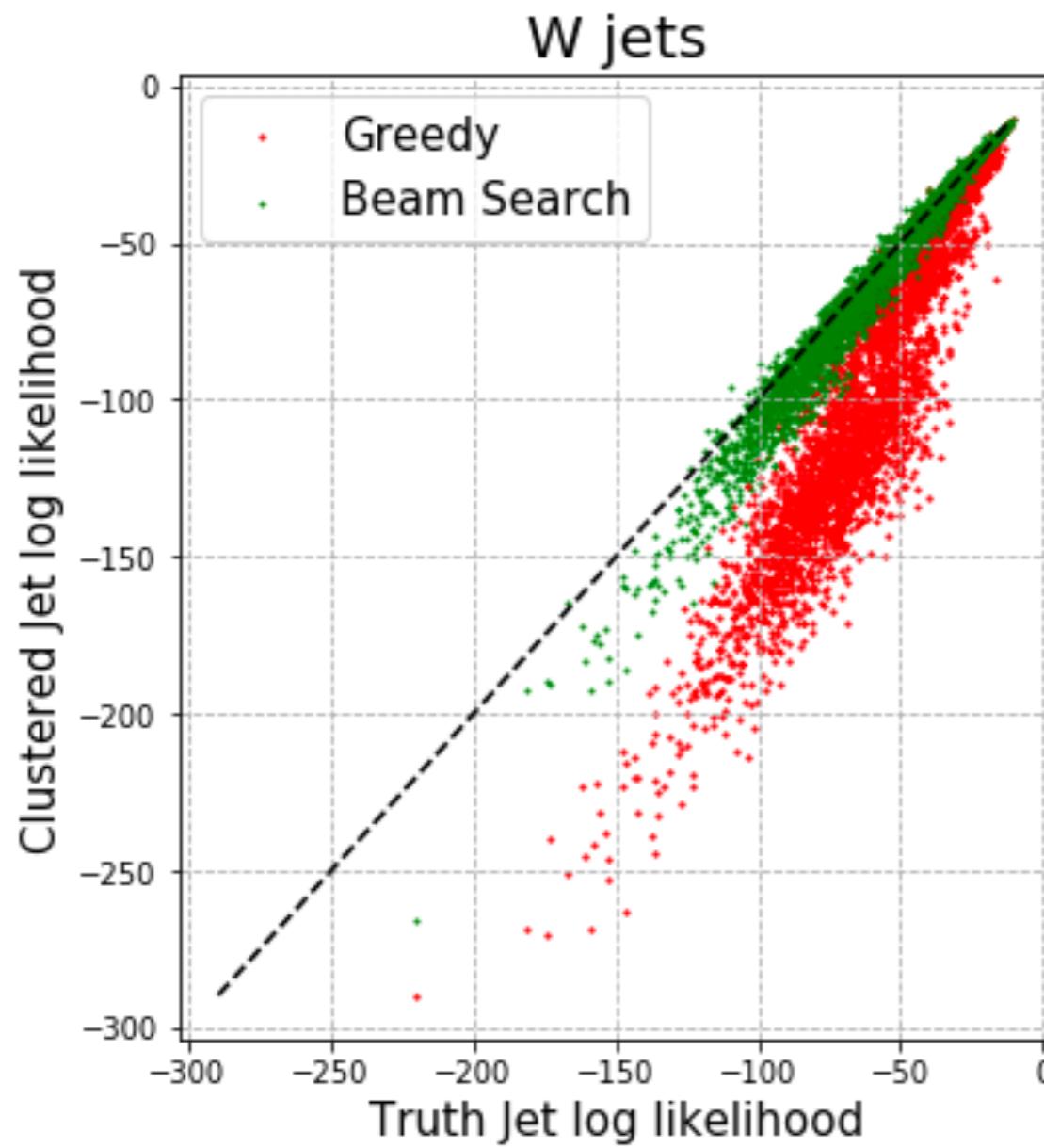


Ginkgo QCD jets

$$\Delta_{\text{root}} \sim f(\Delta | \lambda, \Delta_0) = \frac{\lambda}{\Delta_0} e^{-\frac{\lambda}{\Delta_0} \Delta}$$



# Jets log likelihood



## Mean values:

$$\log p_{\text{truth}} = -45.5 \pm 3.1$$

$$\log p_{\text{BS}} = -46.7 \pm 3.2$$

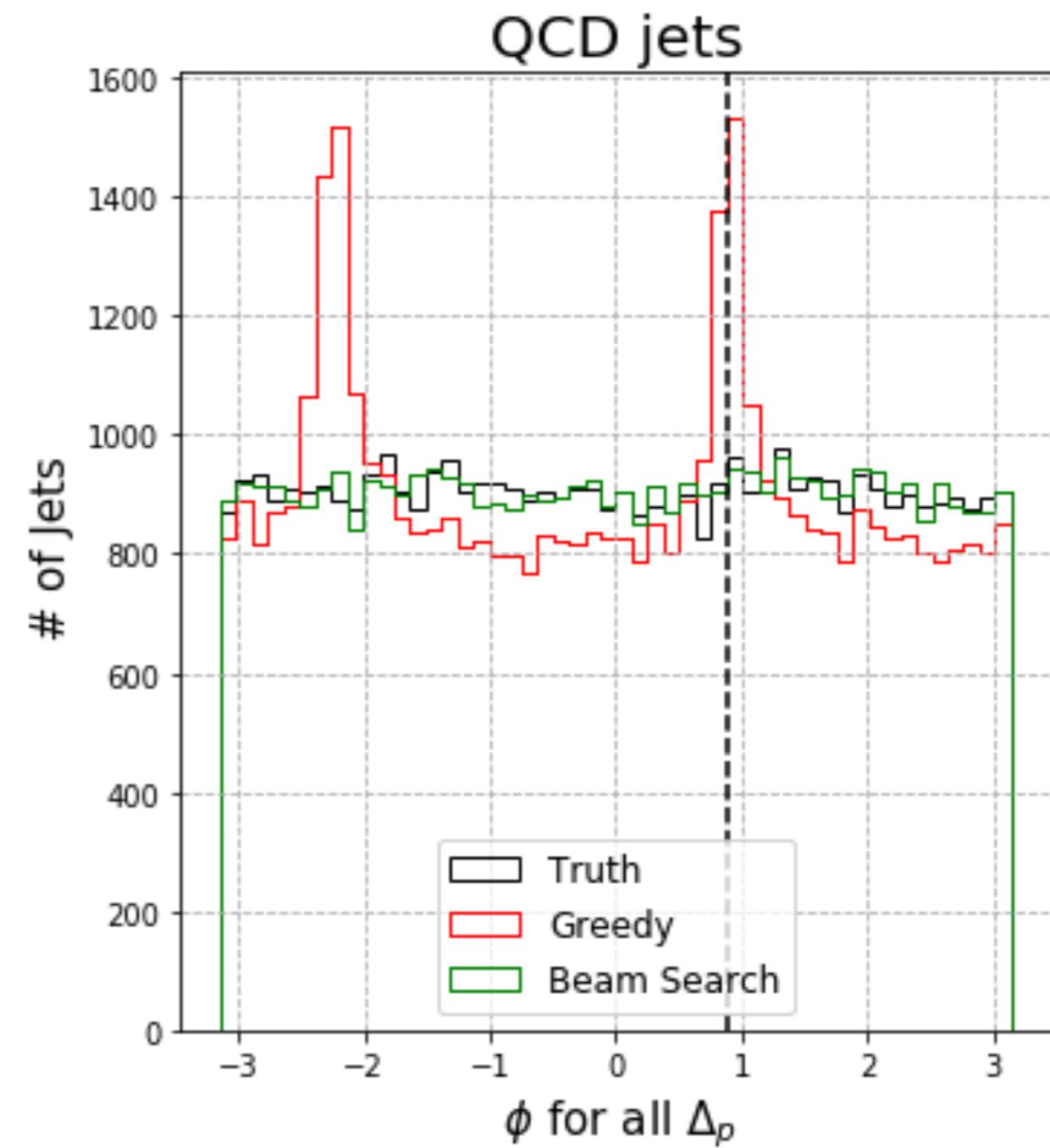
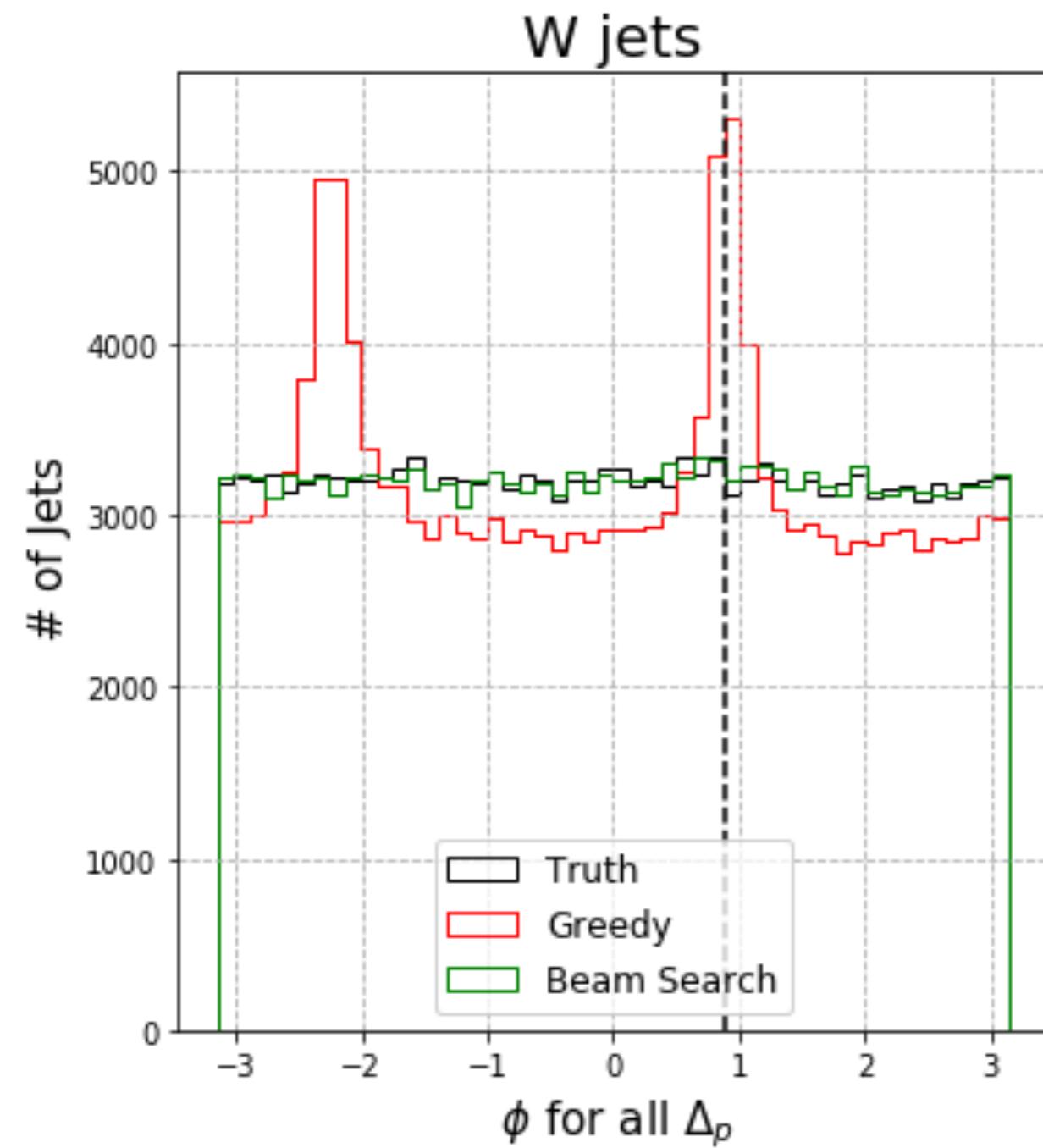
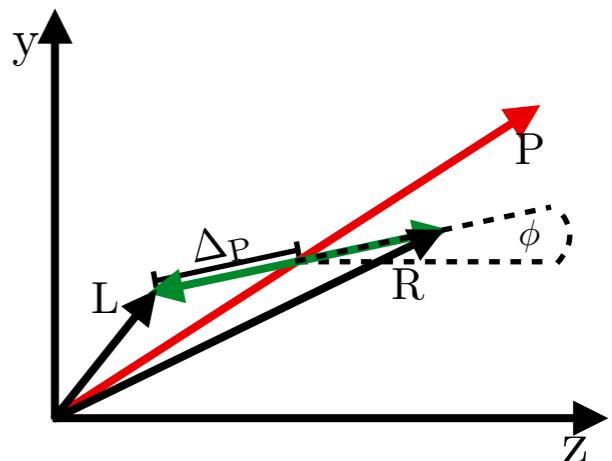
$$\log p_{\text{Greedy}} = -60.9 \pm 5.2$$

$$\log p_{\text{truth}} = -40.8 \pm 2.9$$

$$\log p_{\text{BS}} = -41.4 \pm 3.3$$

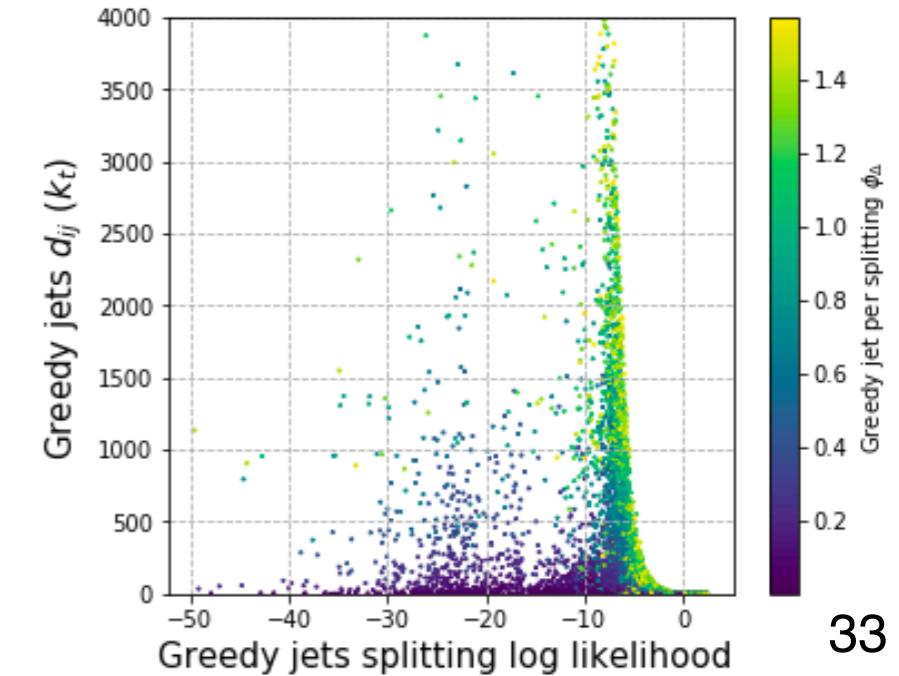
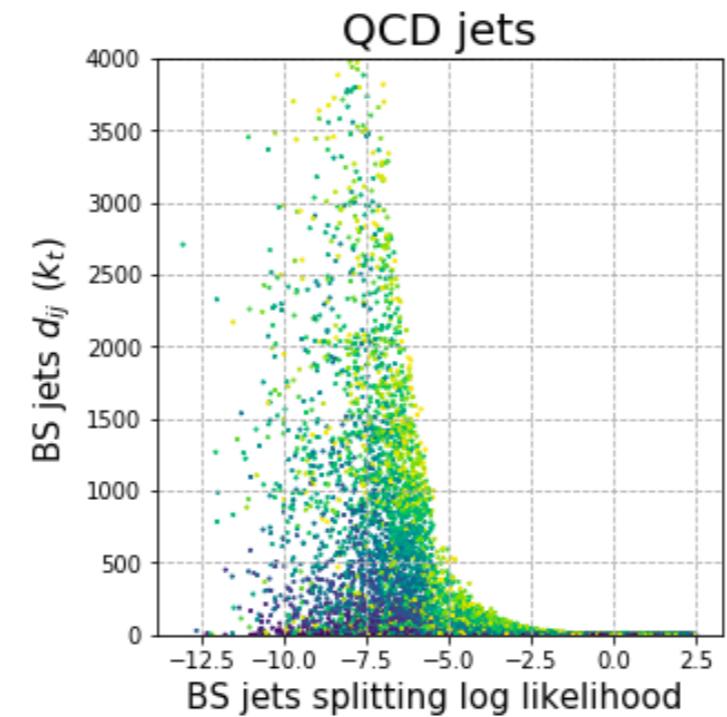
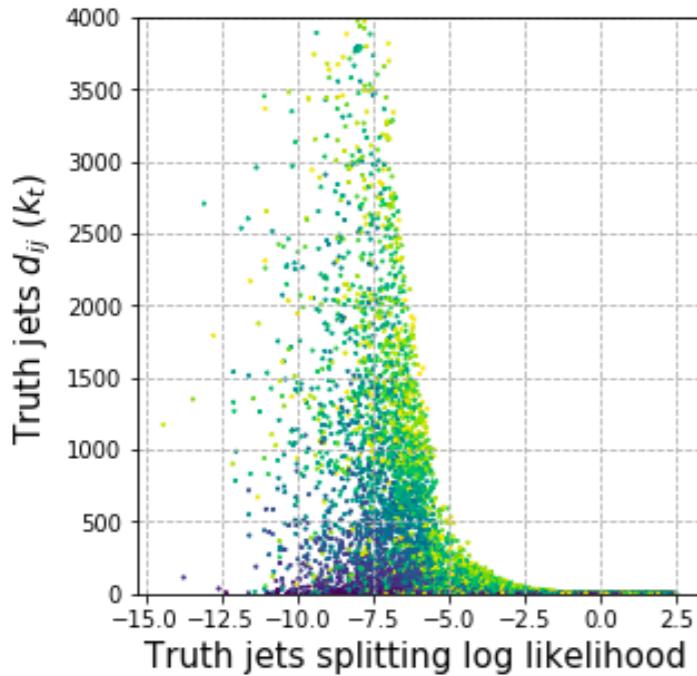
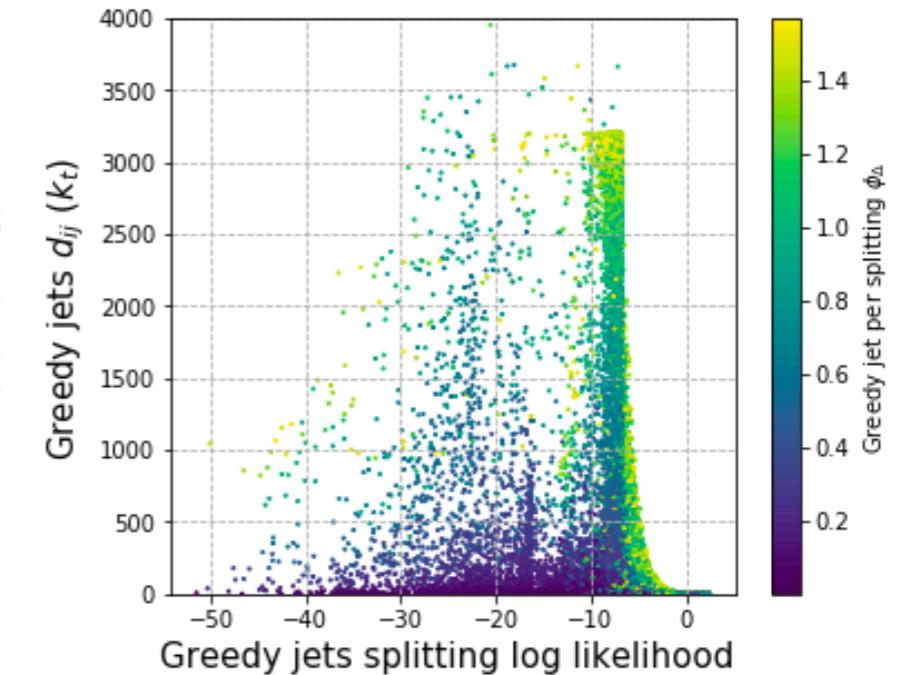
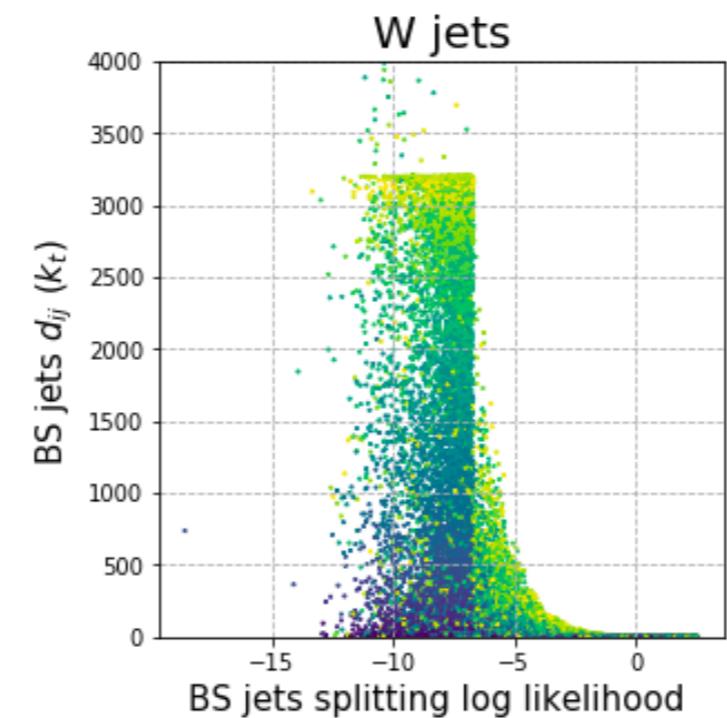
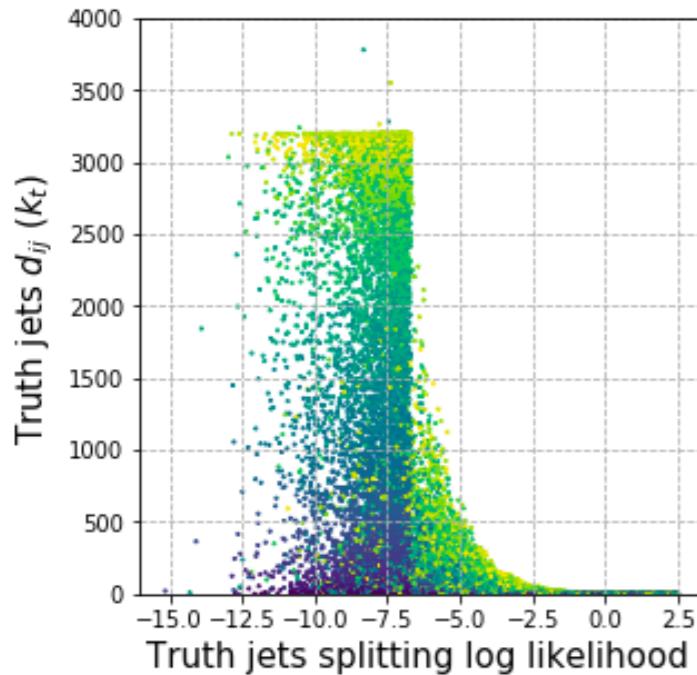
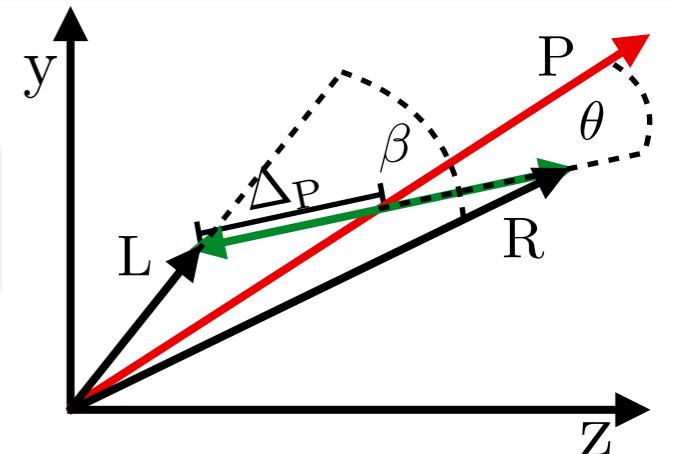
$$\log p_{\text{Greedy}} = -53.5 \pm 5.6$$

# $\phi$ angle



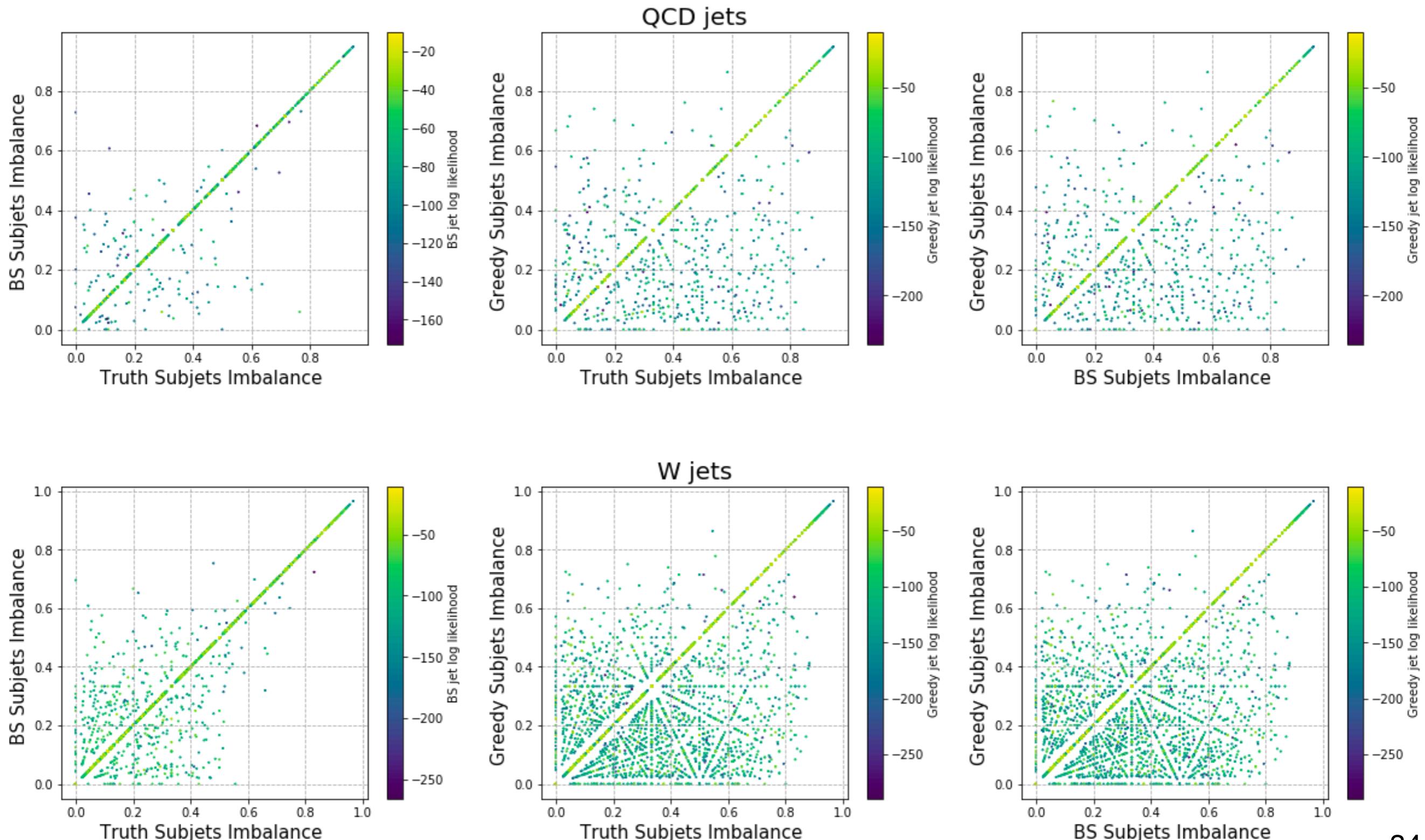
# Splitting log LH vs kt distance metric $d_{ij}$

$$d_{ij} \sim \beta^2$$



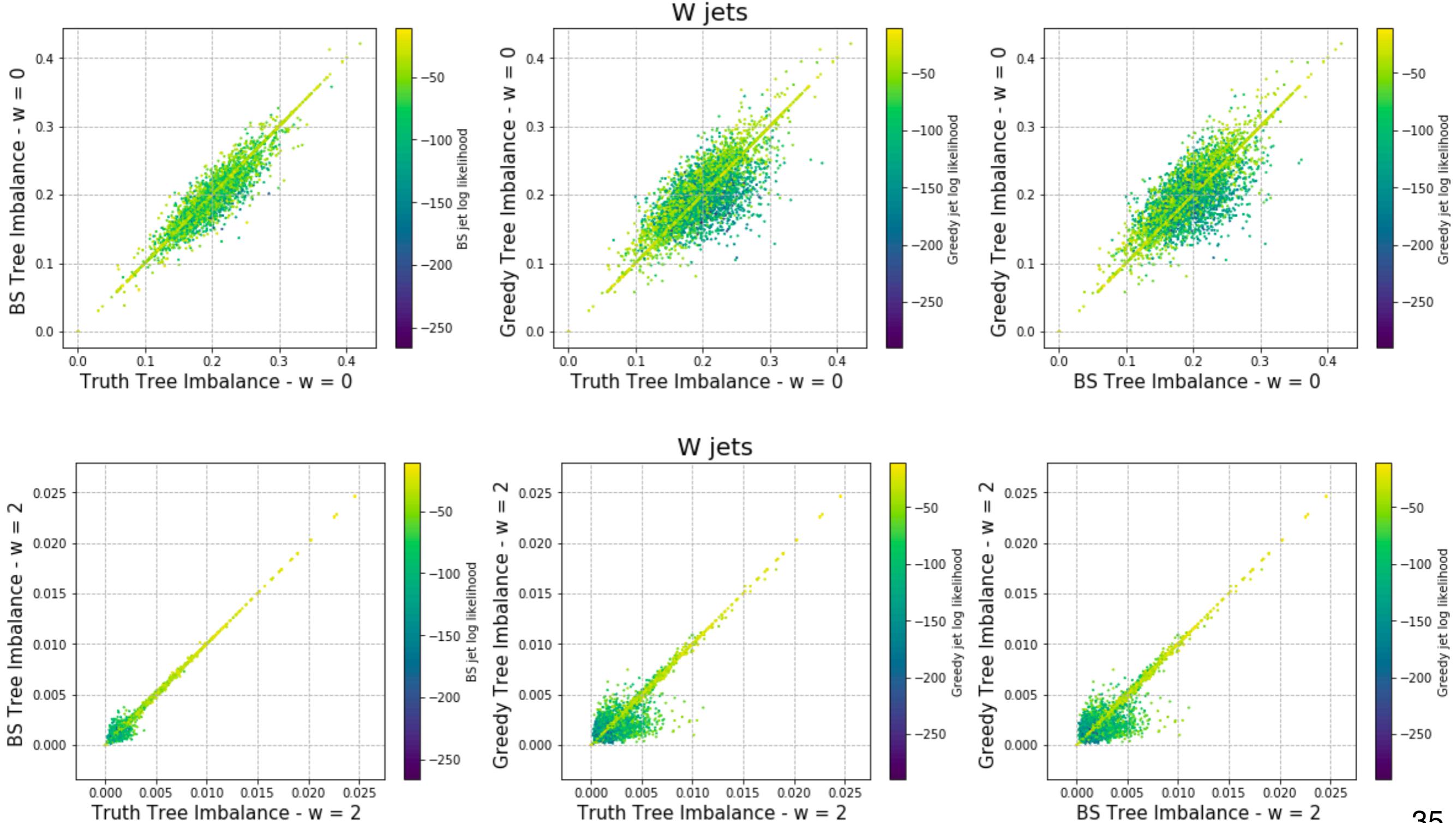
# Subjets imbalance

$$\mathcal{I} = \text{abs}\left(\frac{N_L - N_R}{N_L + N_R}\right)$$



# Tree imbalance

$$\mathcal{T} = \frac{1}{N_{\text{inner}}} \sum_i \exp(-w N_\ell) \mathcal{I}_i$$



# Final remarks and future directions

- Goal: Unification of generation and inference.
- Beam search is strictly better than the greedy algorithm (e.g. kt) in terms of estimating the most likely tree.
- Greedy and beam search algorithms are just starting points. New ideas to study this problem, e.g jet physics implementation of DIORA [A. Drozdov, P. Verga, M. Yadav, M. Iyyer, A. McCallum '19]?
- Study algorithms for sub-trees reconstruction and contamination (soft radiation) removal, e.g reinforcement learning based algorithms.
- Study the posterior distribution  $p(z|x, \theta)$  on clustering histories, conditioned on the observed particles.  
Obtain an estimate from sampling with MCMC or probabilistic programming techniques?
- Could we learn the node splitting from fitting to data and systematically improve the generative model?
- Study metrics to compare binary trees latent structures.

Thanks for your attention!



