Unfolding jet substructure observables with MultiFold in Run 12 200 GeV pp collisions

Hard probes PWG meeting

Youqi Song

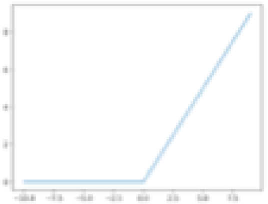
7/21/22

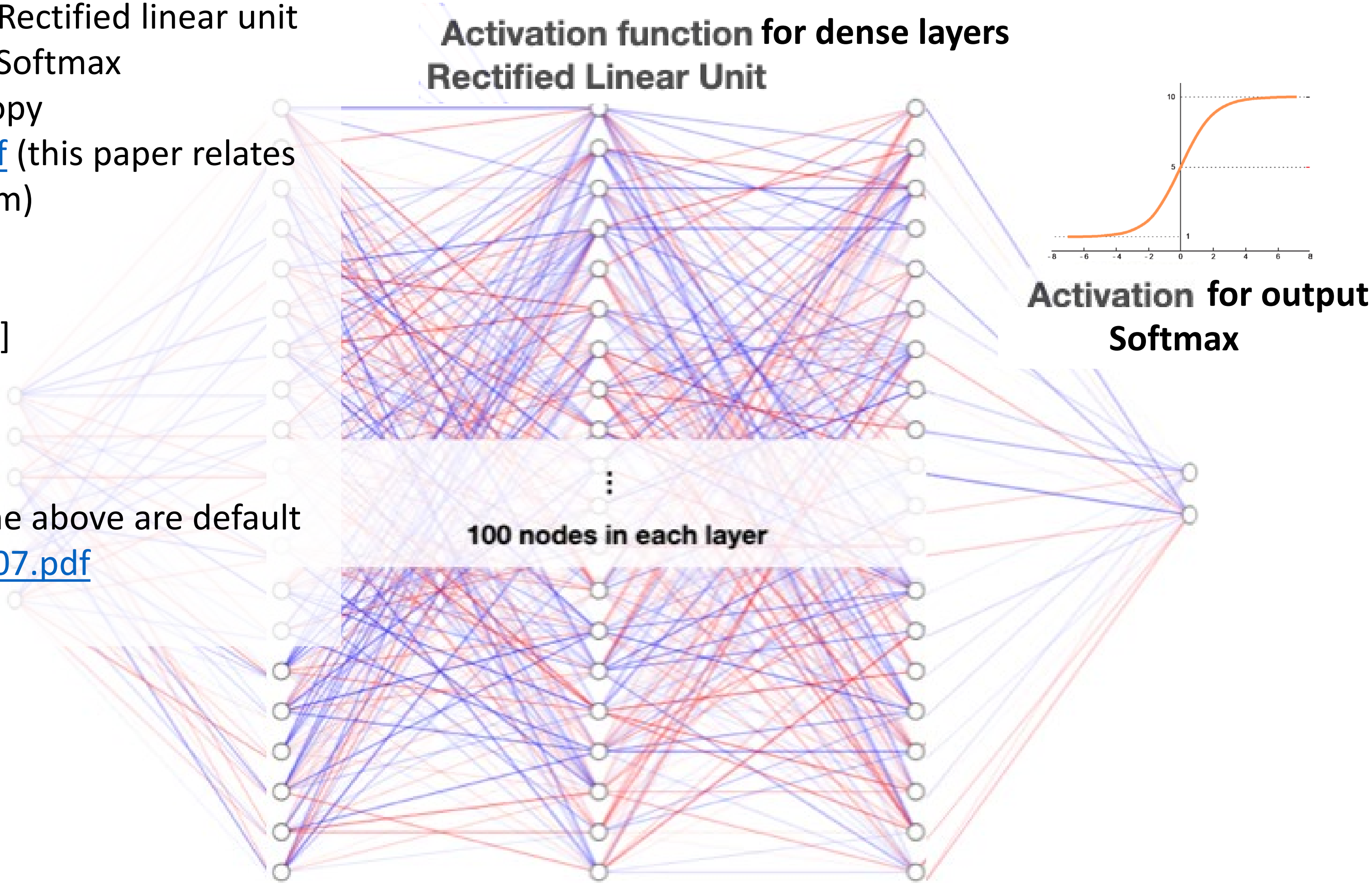
Previous presentation:
https://drupal.star.bnl.gov/STAR/system/files/pwg_meeting_061622-1.pdf

MultiFold introduction – Model

- Architechture: Dense neural network
  https://energyflow.network/docs/archs/#dnn
- Activation function for dense layers: Rectified linear unit
- Activation function for output layer: Softmax
- Loss function: Categorical cross entropy
  https://arxiv.org/pdf/1907.08209.pdf (this paper relates reweighting to categorization problem)
- Optimization algorithm: Adam
  https://arxiv.org/pdf/1412.6980.pdf
- Nodes per dense layer: [100,100,100]
- Output dimension: 2
- Input dimension: 6

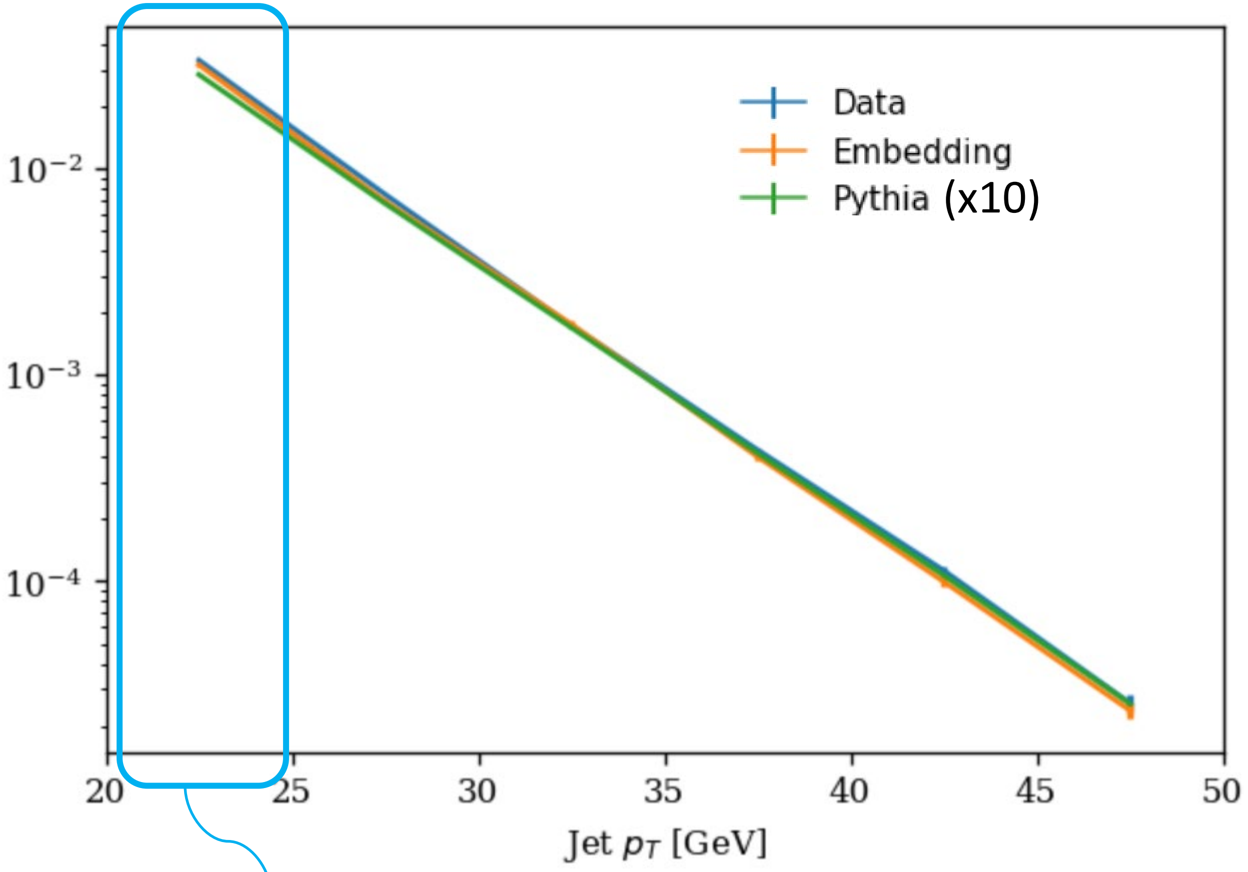
- For more details see backup. All of the above are default from https://arxiv.org/pdf/1911.09107.pdf





Datasets: Run 12 200 GeV pp JP2 triggered

- Data and embedding agree nicely (fake fraction is small)



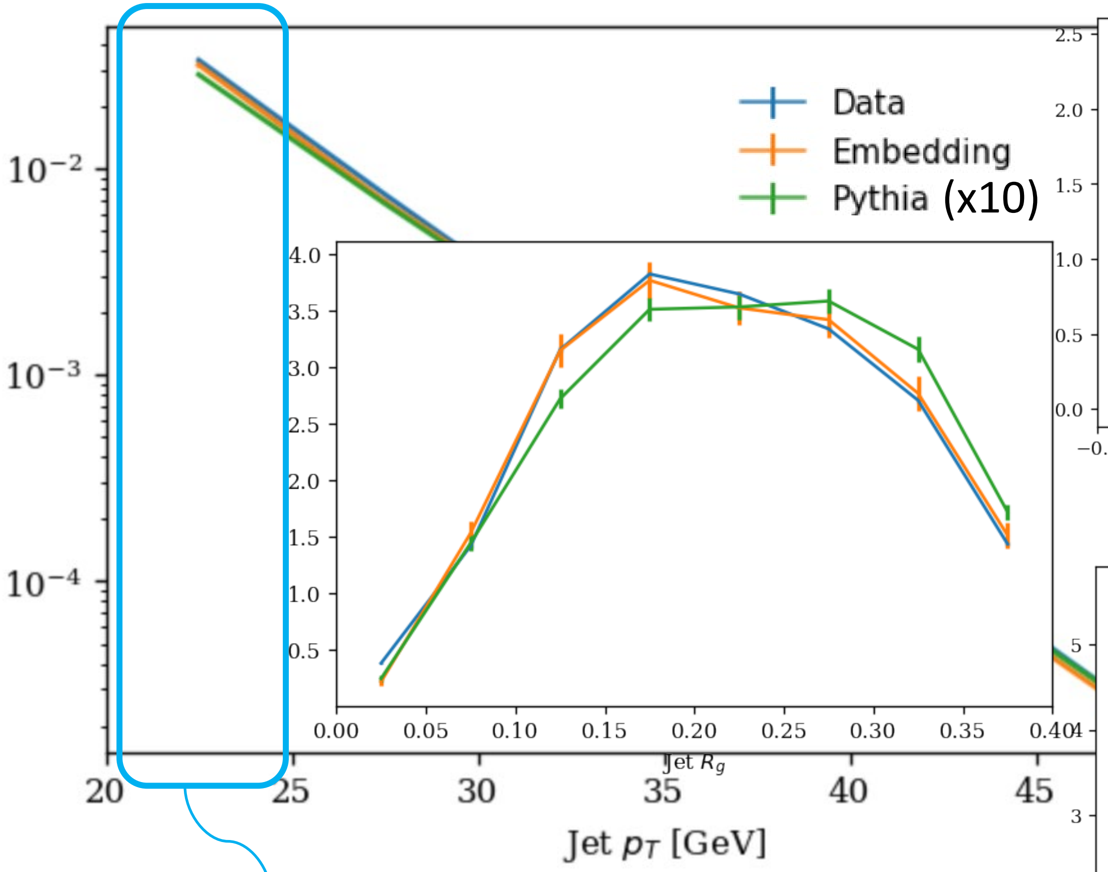
Plot substructure observable distributions for pT of **20-25 GeV**

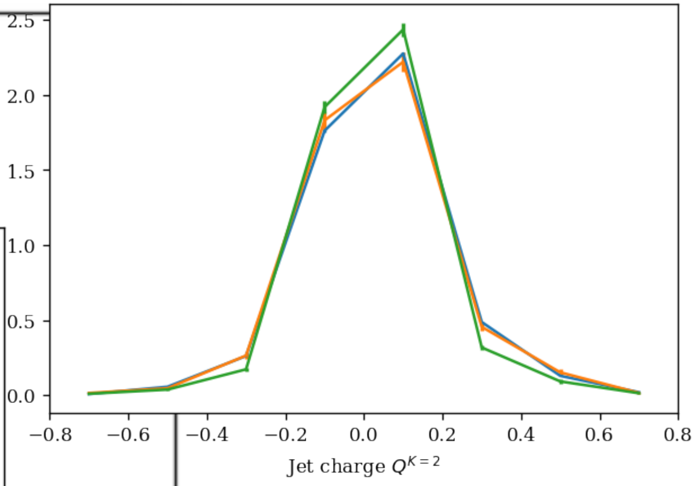
Datasets: Run 12 200 GeV pp JP2 triggered

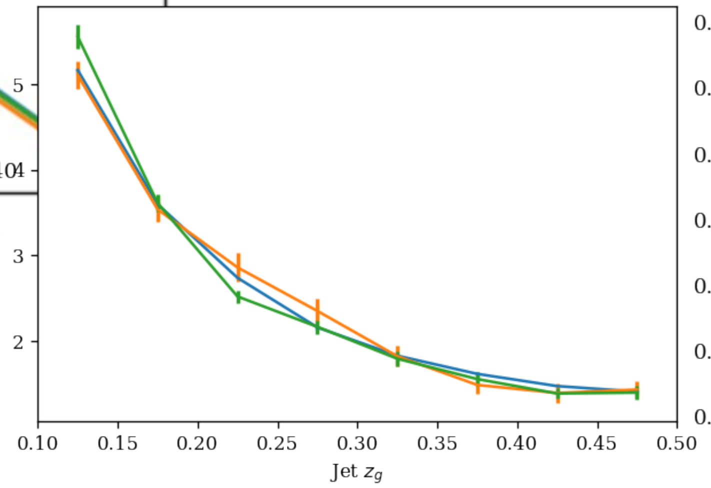
- Data and embedding agree nicely (fake fraction is small)

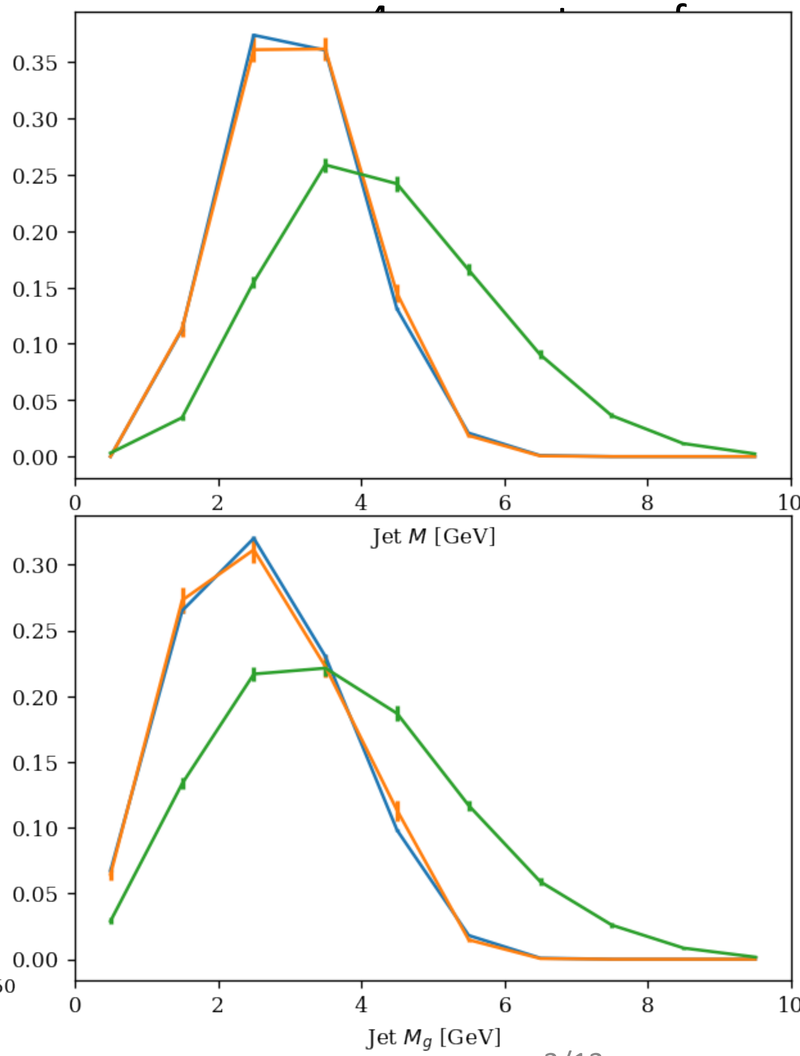
$$Q_J = \frac{1}{(p_{\mathrm{T}\,J})^{\kappa}} \sum_{i \in Tracks} q_i \times (p_{\mathrm{T,i}})^{\kappa}$$

$$M = \left|\sum_{i\in \mathrm{jet}} p_i\right| = \sqrt{E^2 - \boldsymbol{p}^2},$$







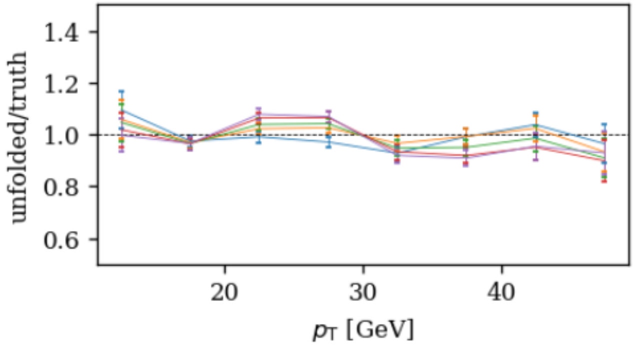


Plot substructure observable distributions for pT of **20-25 GeV**

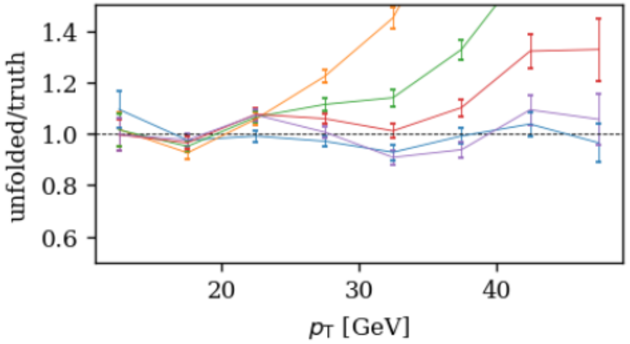
MultiFold closure test

- How many iterations to choose ?

Seed = 0



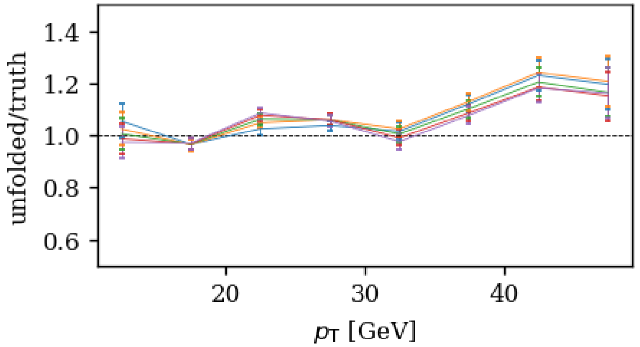
Seed = 1

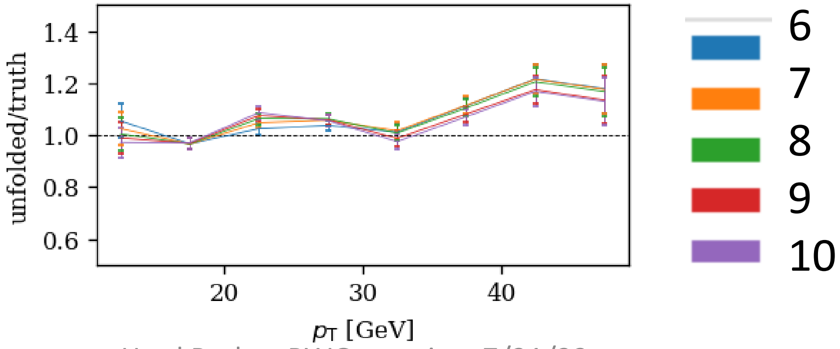


Number of iterations

- See backup for closure for other observables

- Average over 100 seeds



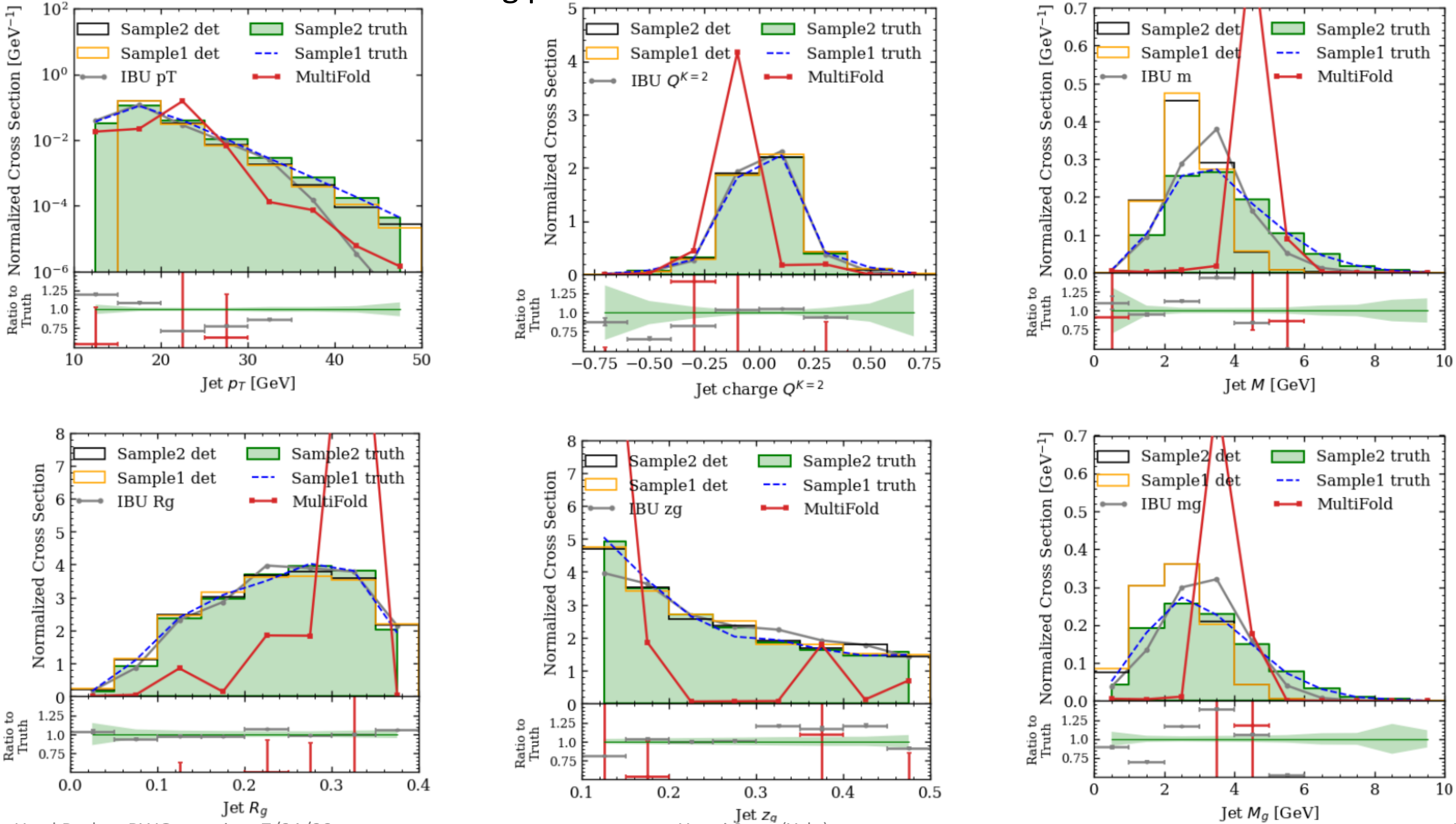


- After averaging over many seeds, unfolded spectrum has a smaller dependence on the number of iteration, and closure is good.

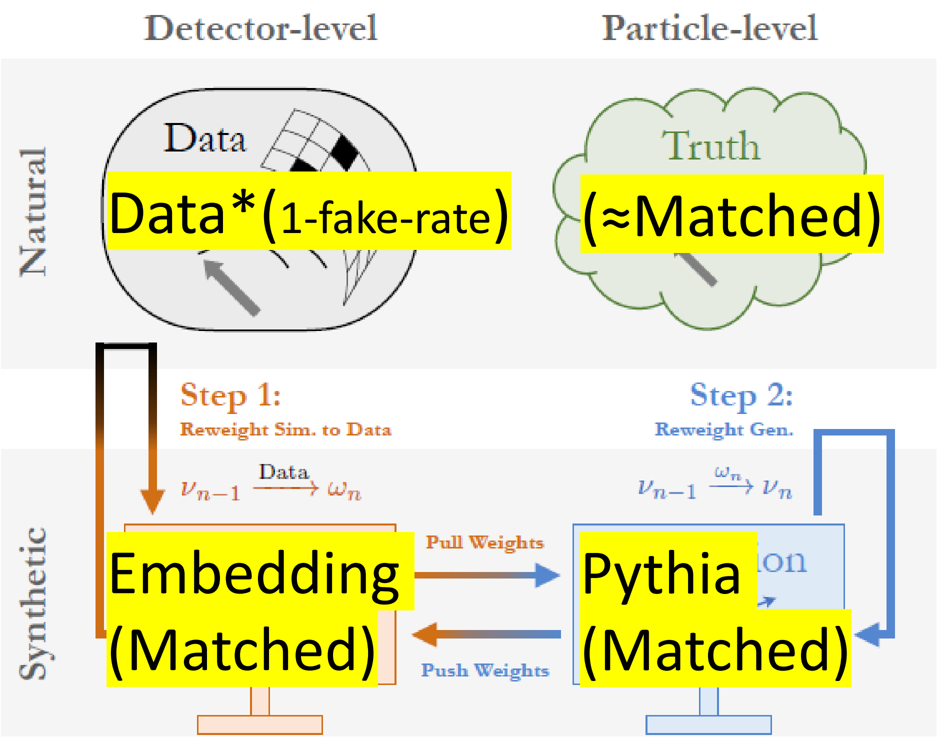
- Anything between 2 and 10 iterations gives reasonable closure. We will see that in data, **variation due to different number of iterations is much smaller than the statistical uncertainty.**

MultiFold closure test

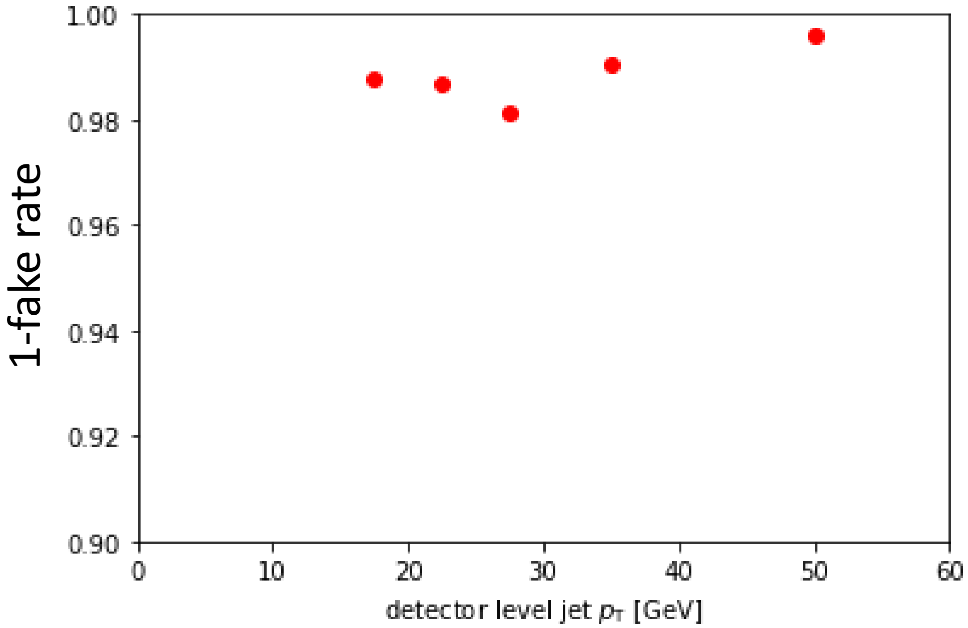
- 20 iterations fails – there is a breaking point



MultiFold on data - Method

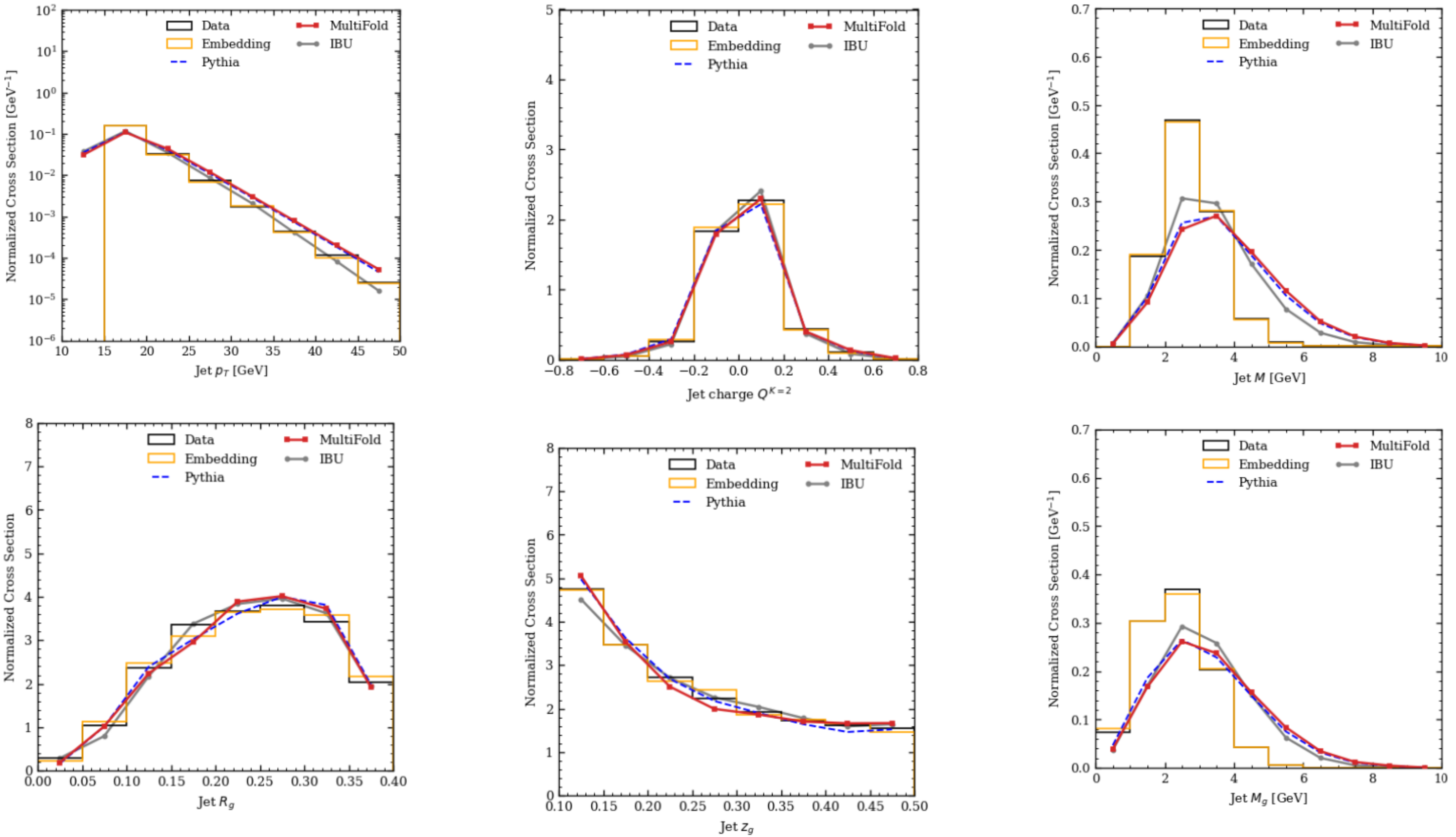


- Put in (1-fake rate) as initial weights for data
  - Checked that fake rate is roughly independent of distribution of other observables (see backup)
- Pythia and embedding samples are matched jets only
- Deal with misses after unfolding



MultiFold on data

➢ MultiFolded spectrum roughly agrees with Pythia (truth).

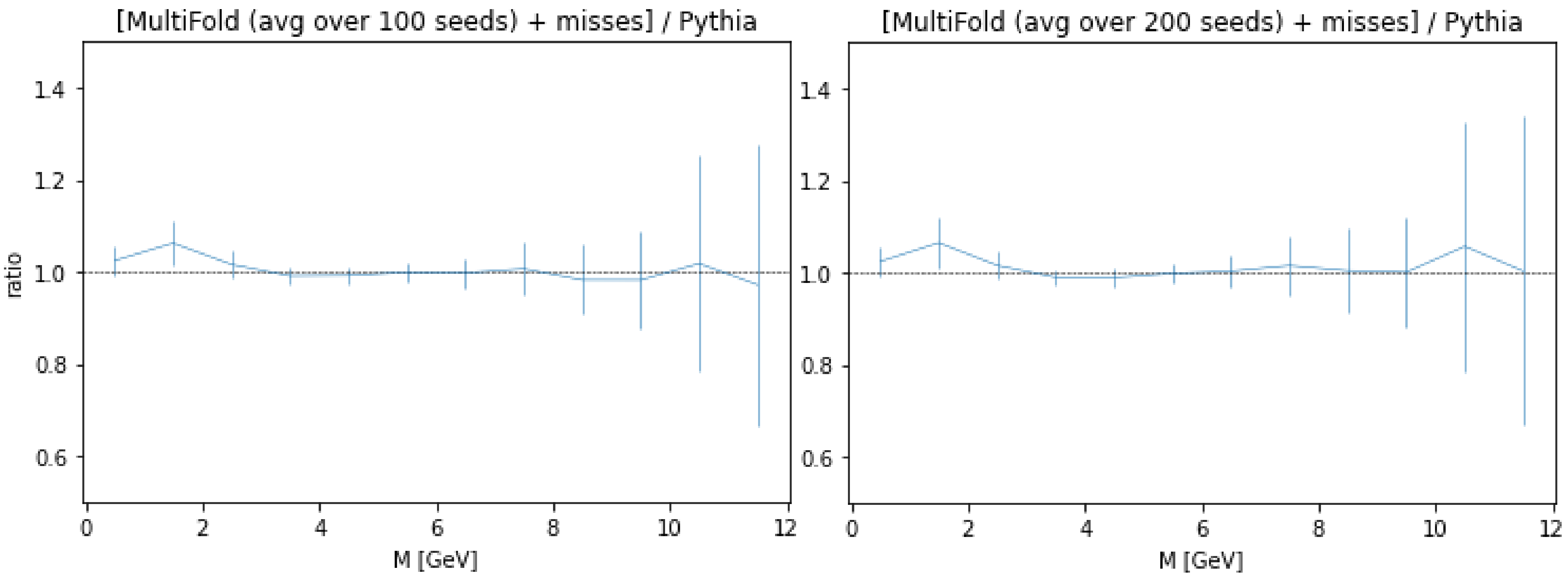


- Pythia and embedding samples are matched jets only
- Result is unbinned (bins were chosen just for plotting)
- Data shown is raw data, but fake rate is taken into account for MultiFold
- Substructure observables NOT binned in pT
- **Averaged over 100 different random seeds**

itnum = 4
patience1 = 50
patience2 = 50
valnum = 0.2
batch_size1 = 50000
batch_size2 = 10000

MultiFold on data

➢ Averaging over more seeds doesn't change the result significantly.



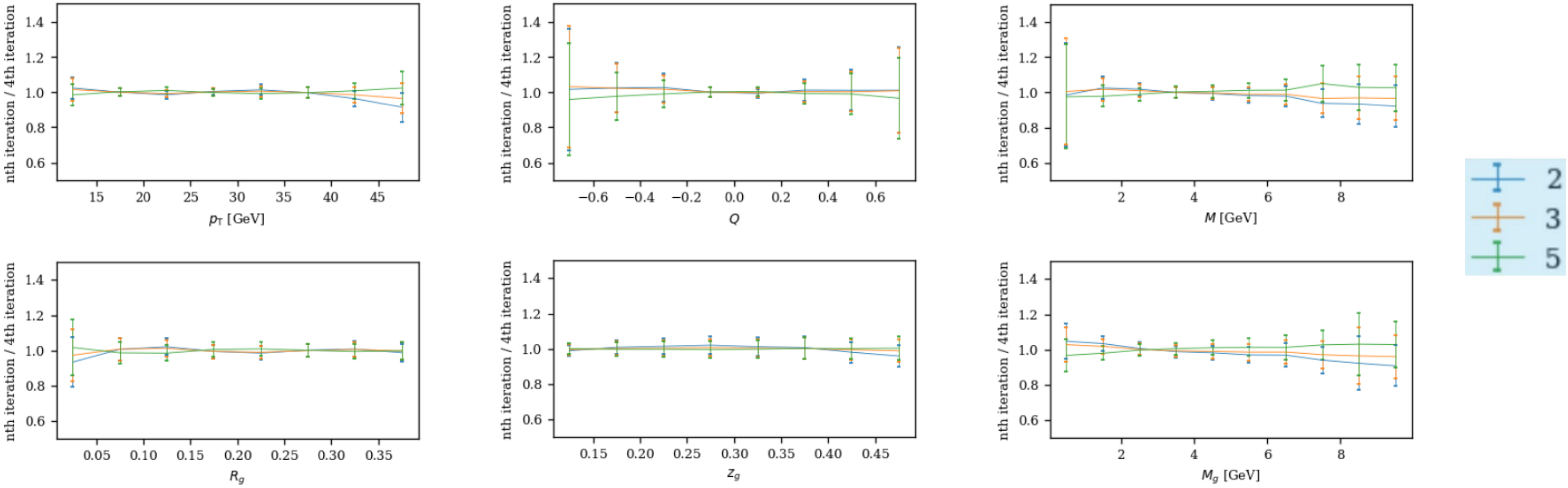
- Pythia and embedding samples are matched jets only
- Result is unbinned (bins were chosen just for plotting)
- Data shown is raw data, but fake rate is taken into account for MultiFold
- Substructure observables NOT binned in pT
- **Averaged over 100 different random seeds**

Error bars only include standard deviations from varying seeds (to be included as systematics).
Other systematic uncertainties will be shown on later slides.

itnum = 4
patience1 = 50
patience2 = 50
valnum = 0.2
batch_size1 = 50000
batch_size2 = 10000

MultiFold on data

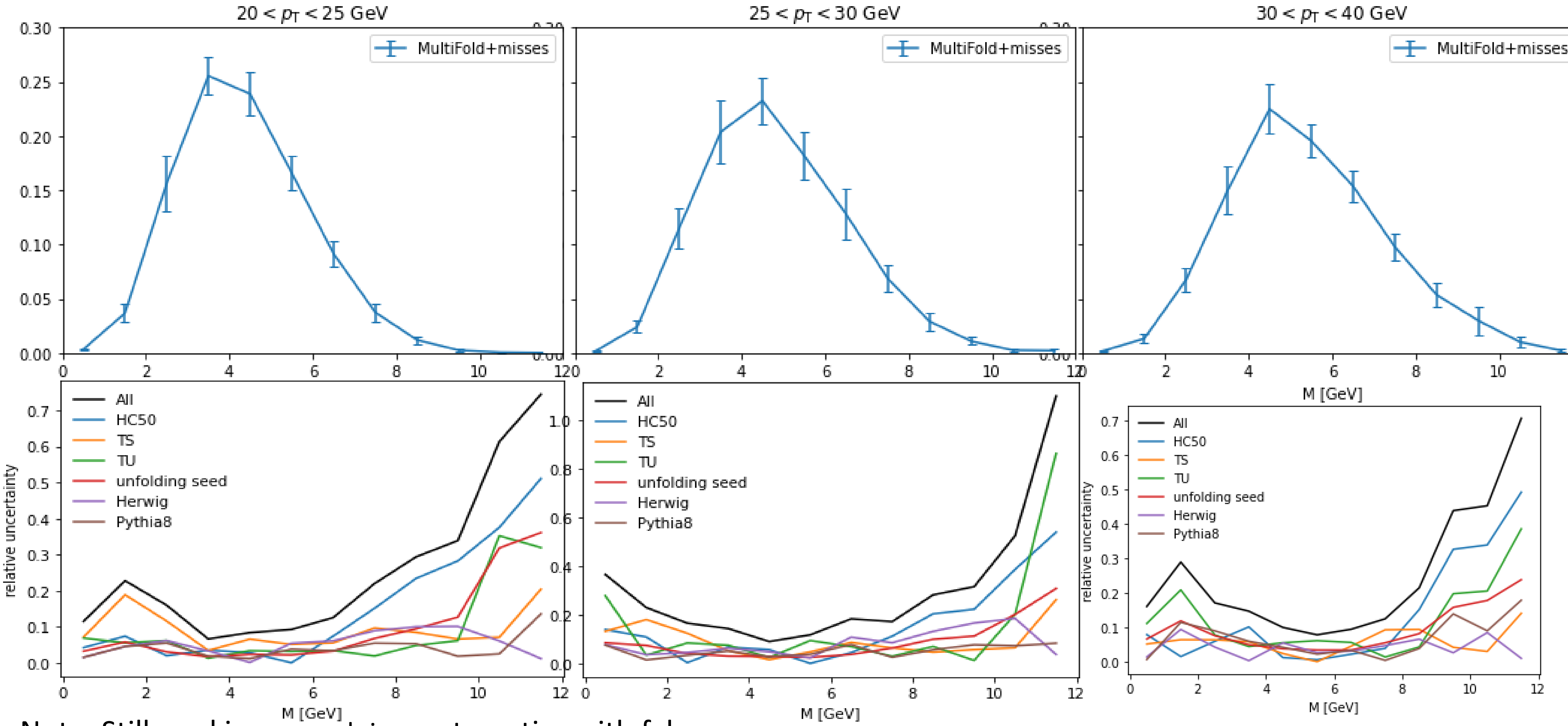
- If averaged over 100 seeds, **<u>variation due to different number of iterations is much smaller than the statistical uncertainty</u>**. We will choose 4 iterations as our “nominal” value. There will not be systematic uncertainty associated with number of iterations.



Full spectra for each pT bin
= efficiency * MultiFolded spectrum + (1-eff) * Pythia distribution for misses

Systematics
- Hadronic correction 100% -> 50%
- Tower scale +3.8%
- Tracking uncertainty -4%
- Unfolding prior pT weights -> Herwig/Pythia8
- Unfolding seed

Full spectrum – jet mass

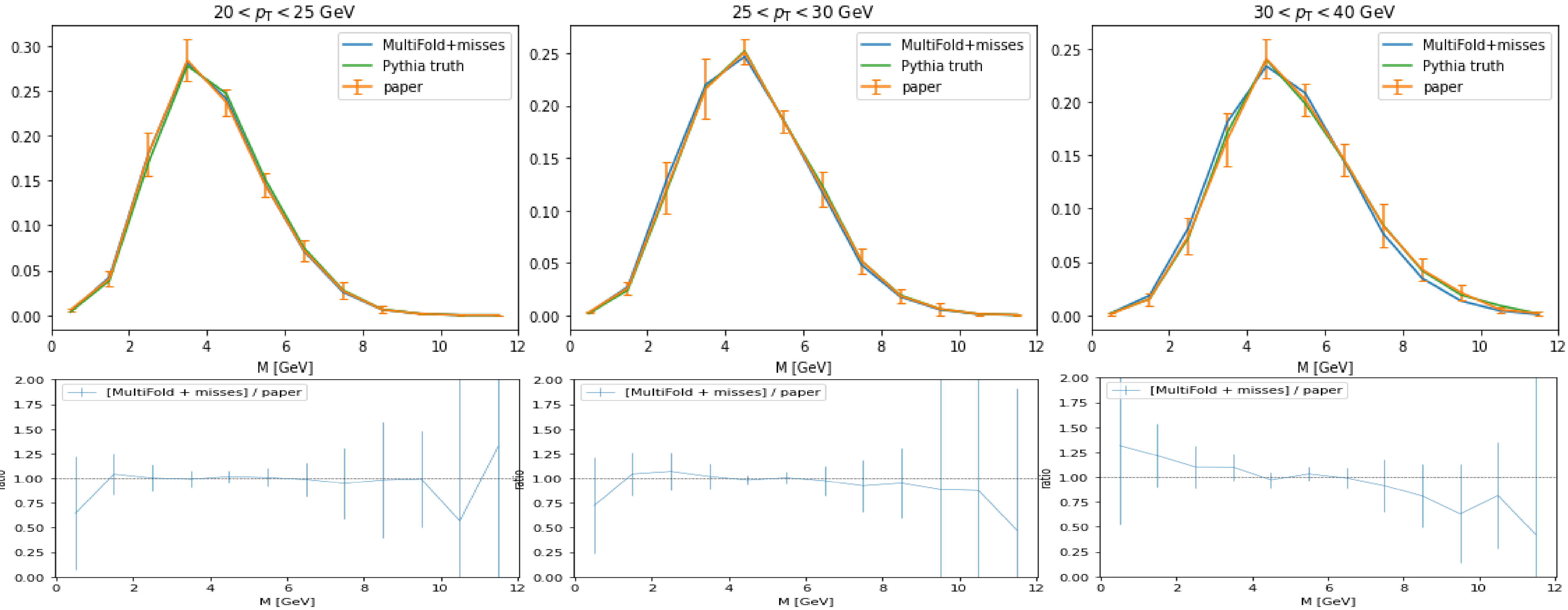


Note: Still working on redoing systematics with fakes

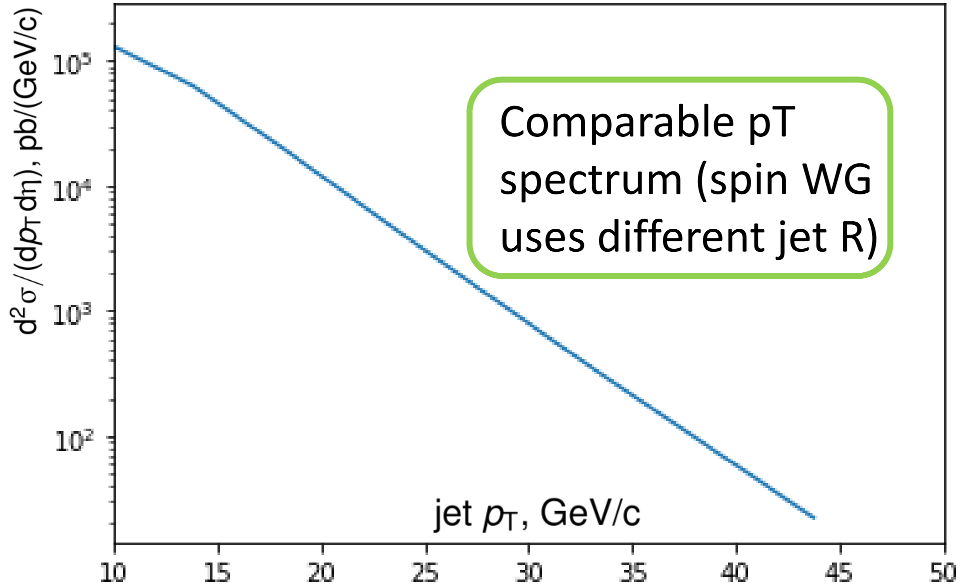
Full spectrum comparison – jet mass

- Since Isaac used a different assignment of particle eta, we have different jet populations. Run the whole procedure (except I didn't average over 100 seeds and didn't put in systematics) with the same eta assignment as Isaac:

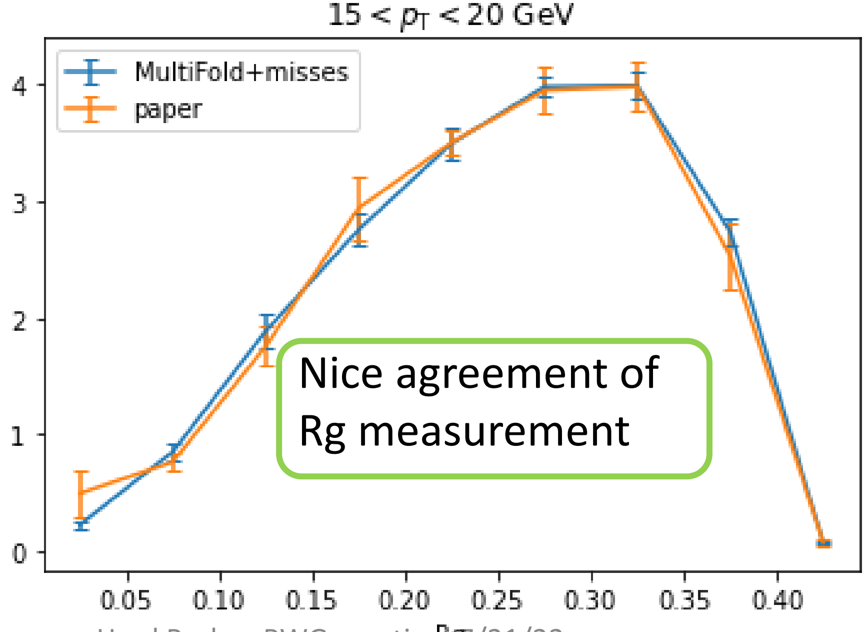
➢ Multifold result agrees with Isaac's result!



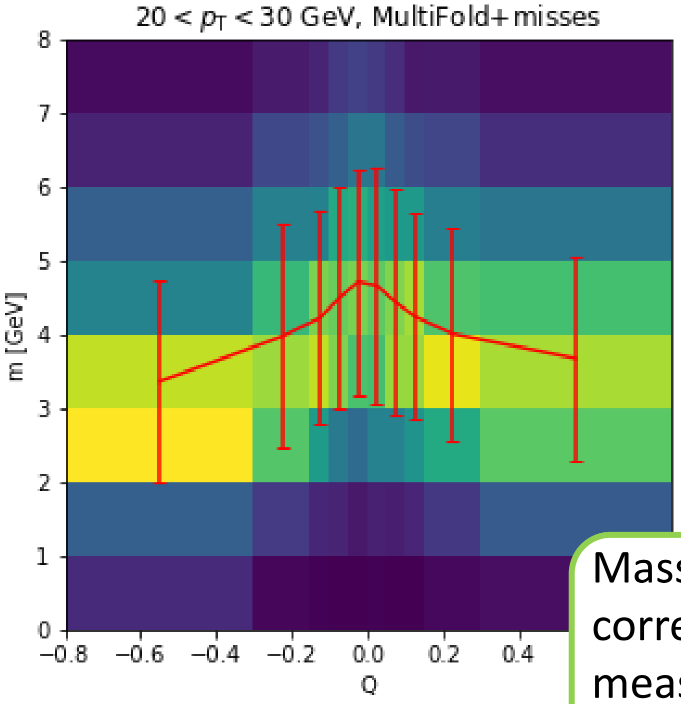
Coming up next week / in 2 weeks

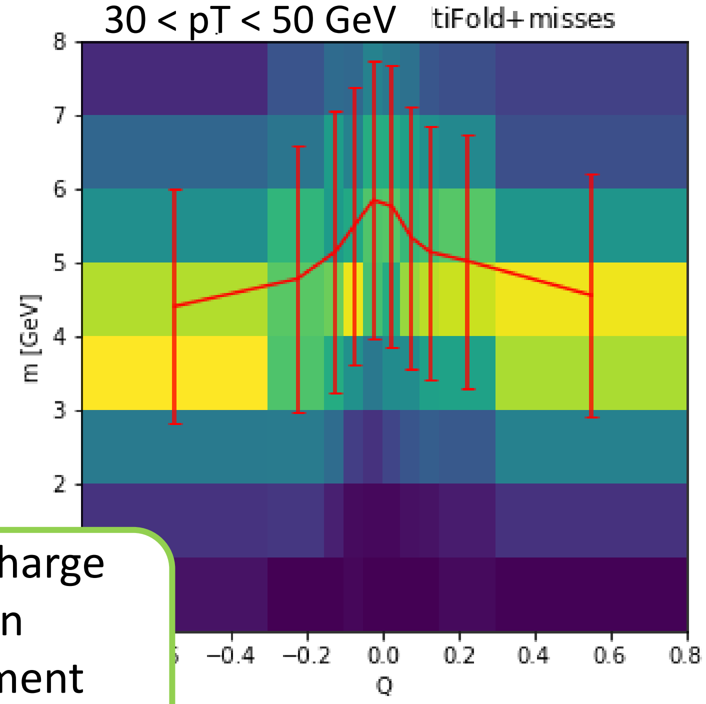


Comparable pT spectrum (spin WG uses different jet R)

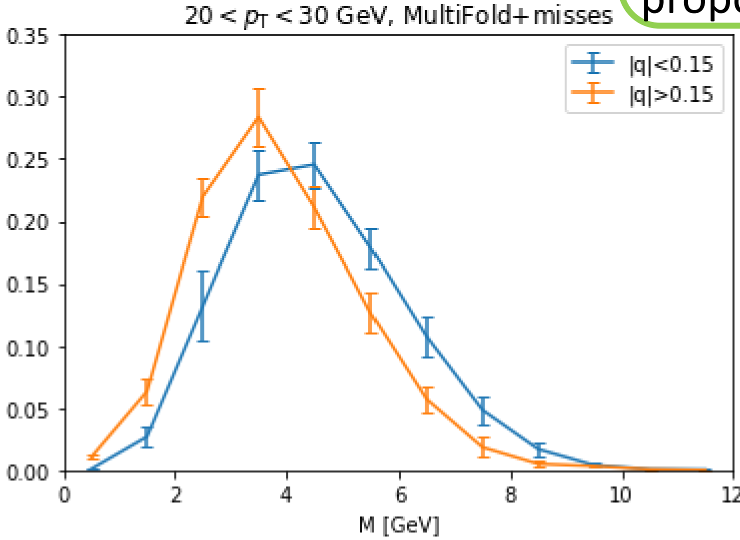


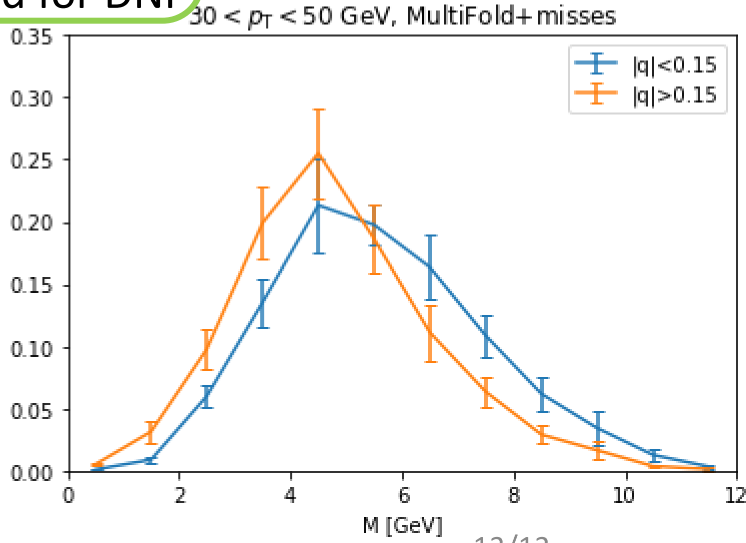
Nice agreement of Rg measurement





Mass vs charge correlation measurement proposed for DNP





Backup

MultiFold introduction - Motivation

- We want to study the correlations between jet substructure observables for jets in pp
  - pT
  - $Q_J = \frac{1}{(p_{T\,J})^{\kappa}} \sum_{i \in Tracks} q_i \times (p_{T,i})^{\kappa}$
  - $M = \left|\sum_{i\in \text{jet}} p_i\right| = \sqrt{E^2 - \boldsymbol{p}^2},$

4-momentum of the constituent i

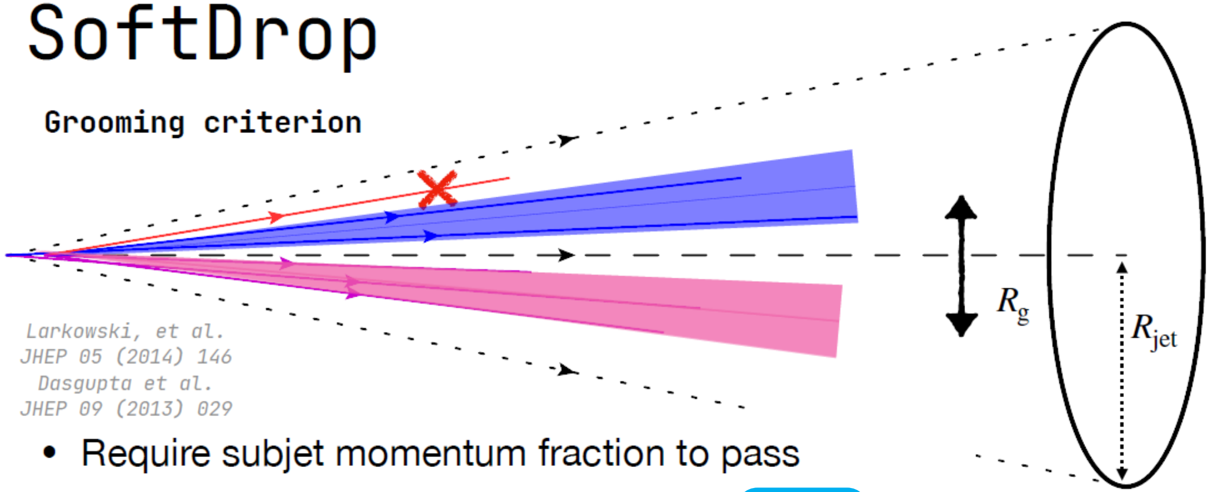
  - Rg
  - Zg
  - Mg

Groomed jet observables

Slide from Raghav:

SoftDrop

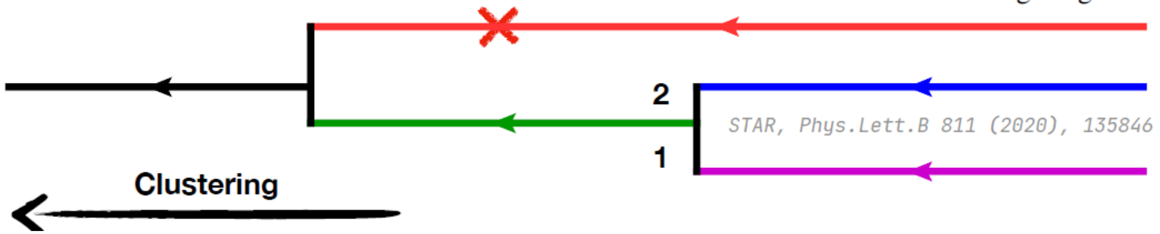
Grooming criterion



- Require subjet momentum fraction to pass

$$z_g = \frac{\min(p_{T,1}, p_{T,2})}{p_{T,1} + p_{T,2}} > z_{cut}(R_g/R_{jet})^{\beta} \qquad \boxed{\begin{array}{l} z_{cut} = 0.1 \\ \beta = 0 \end{array}}$$

- With the two surviving branches (first hard split) - we define observables that characterize jet substructure z_g, R_g



MultiFold introduction – Method

- How did we turn unfolding into a classification problem?
  - In one equation: $$L[(w,X),(w',X')](x) = \frac{p_{(w,X)}(x)}{p_{(w',X')}(x)} \approx f(x)/(1-f(x))$$

This is by definition the inverse of the response matrix*

where f(x) is a neural network and trained with the binary cross-entropy loss (loss function for a categorization problem)**

*: see https://arxiv.org/pdf/1911.09107.pdf, specifically this section:

to the unbinned, full phase space. A key concept for this approach is the likelihood ratio:

$$L[(w,X),(w',X')](x) = \frac{p_{(w,X)}(x)}{p_{(w',X')}(x)}, \tag{3}$$

where $p_{(w,X)}$ is the probability density of x estimated from empirical weights w and samples X. The function $L[(w,X),(w',X')](x)$ can be approximated using a classifier trained to distinguish (w,X) from (w',X'). This property has been successfully exploited using neural networks for full phase-space Monte Carlo reweighting and parameter estimation [18, 22–26]. Here, we use neural network classifiers to iteratively reweight the particle- and detector-level Monte Carlo weights, resulting in an unfolding procedure.

**: see https://arxiv.org/pdf/1907.08209.pdf, specifically this section:

The first ingredient to the full phase-space reweighting procedure is a prescription to derive event weights. Consider two simulations that describe the same phase space Ω and are described by probability densities $p_0(x)$ and $p_1(x)$, for $x \in \Omega$. Assuming that p_0 and p_1 have the same support[1], the function $w(x) = p_0(x)/p_1(x)$ is the ideal per-event weight to morph the second simulation into the first one. A key observation made by multiple groups in the past is that w can be well-approximated by training a machine learning classifier to distinguish the two simulations. For example, let $f(x)$ be a neural network and trained with the binary cross-entropy loss:

$$\mathrm{loss}(f(x)) = -\sum_{i\in\mathbf{0}} \log f(x_i) - \sum_{i\in\mathbf{1}} \log(1-f(x_i)), \tag{1}$$

where $\mathbf{0}$ and $\mathbf{1}$ represent sets of examples from the two simulations. Then a well-known result is that[2], $f(x)/(1-f(x)) \approx p_0(x)/p_1(x)$. The benefit of parameterizing f as a neural network is that deep learning can readily analyze all of Ω, which was not possible with shallow learning attempts with a similar statistical foundation. The closest attempt to a full phase space approach

MultiFold introduction - Method

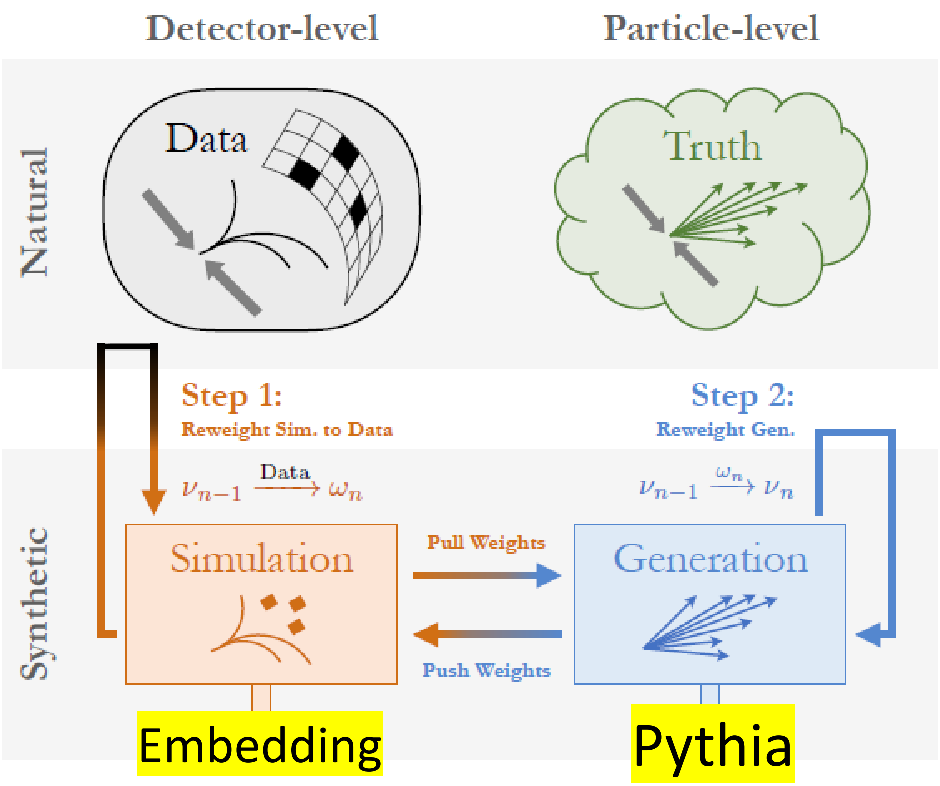


FIG. 1. An illustration of OmniFold, applied to a set of synthetic and natural data. As a first step, starting from prior weights ν_0, the detector-level synthetic data ("simulation") is reweighted to match the detector-level natural data (simply "data"). These weights ω_1 are pulled back to induce weights on the particle-level synthetic data ("generation"). As a second step, the initial generation is reweighted to match the new weighted generation. The resulting weights ν_1 are pushed forward to induce a new simulation, and the process is iterated.

MultiFold introduction - Method

- Iteration 1, Step 1:

Data	Embedding

Training	Validation

Run model.fit() on Training

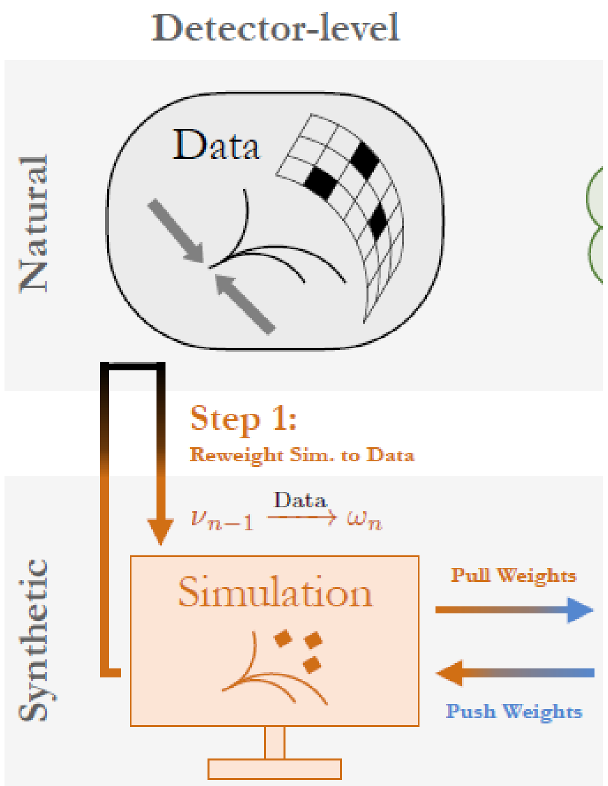
Validation: Decides when training should stop

Run model.predict() on Embedding. Obtain weights.

Weights corresponding to $\boxed{\text{Embedding}}$ are pulled to step 2.

$$L[(w,X),(w',X')](x) = \frac{p_{(w,X)}(x)}{p_{(w',X')}(x)}$$

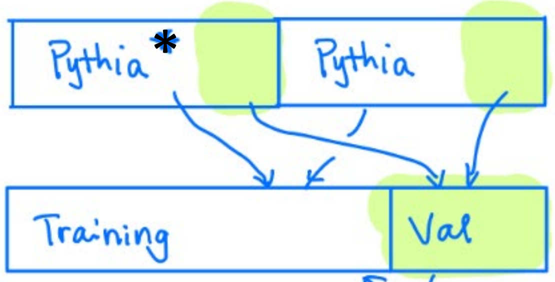
$$\approx f(x)/(1-f(x))$$



- Note: model.predit() returns an array of f(x) (each element is a jet). Weights pulled are f(x)/(1-f(x))

MultiFold introduction - Method

- Iteration 1, Step 2:



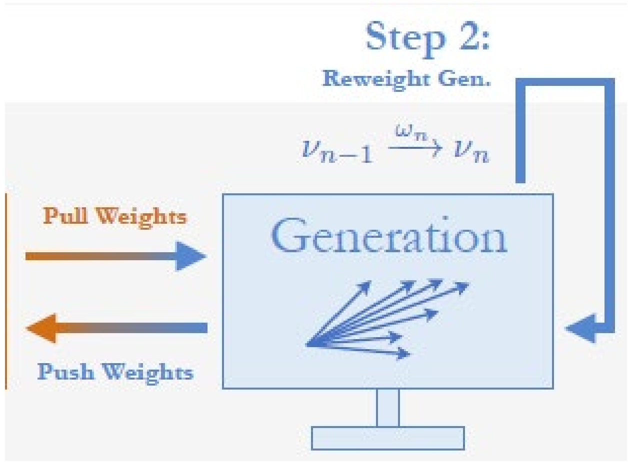
Run model.fit() on

Decides when training should stop

Run model.predict() on . Obtain weights.

Weights corresponding to Pythia are pushed to step 1.

*: (With weights pulled from step 1 of iteration 1)



MultiFold introduction - Method

- Iteration n, Step 1:

Data	Embedding**
Training	Validation

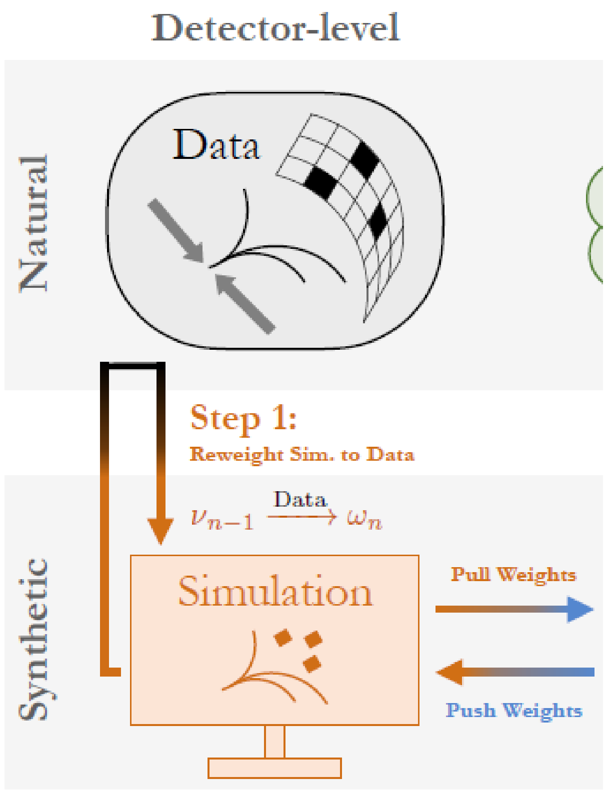
Run model.fit() on Training

Validation: Decides when training should stop

Run model.predict() on Data + Embedding**. Obtain weights.

Weights corresponding to $\boxed{\text{Embedding**}}$ are pulled to step 2.

**: (With weights pushed from step 2 of iteration (n-1))



MultiFold introduction - Method

- Iteration n, Step 2:

Pythia* | Pythia**

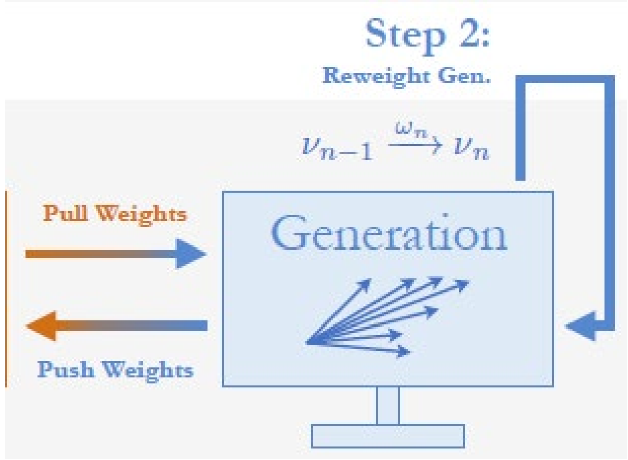
Training | Val

Run model.fit() on Training

Val: Decides when training should stop

Run model.predict() on Val. Obtain weights.

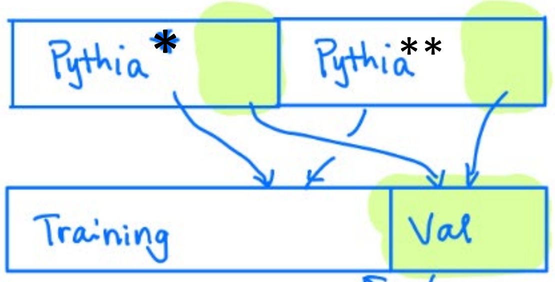
Weights corresponding to Pythia** are pushed to step 1.



*: (With weights pulled from step 1 of iteration n)
**: (With weights pushed from step 2 of iteration (n-1))

MultiFold introduction - Method

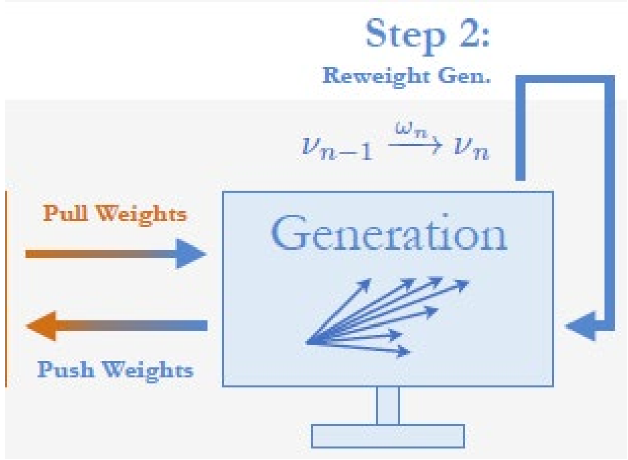
- Iteration n, Step 2:



Run model.fit() on ↖ ↳ Decides when training should stop

Run model.predict() on ↖. Obtain weights.

Weights corresponding to Pythia** are pushed to step 1 or taken as the final weights if this is the last iteration



*: (With weights pulled from step 1 of iteration n)
**: (With weights pushed from step 2 of iteration (n-1))

MultiFold introduction - Method

- Output
  - Result is unbinned (bins were chosen just for plotting)
  - We get the correlation between observables for free

	weight	pt	q	rg	zg	m	mg
0	0.001282	12.478762	-0.340979	0.116877	0.363053	2.130296	1.754463
1	0.000711	20.699182	0.060520	0.124359	0.482597	3.381928	1.409338
2	0.001819	14.479642	0.049692	0.157490	0.478144	3.463364	3.463364
3	0.000652	14.214460	0.018048	0.278783	0.356131	4.713497	4.713497
4	0.001041	16.045917	0.257764	0.316269	0.109652	2.585799	2.585799
...	...	...	...	...	...	...	...
33707	0.001054	16.891453	-0.108731	0.251257	0.153995	2.508725	2.508725
33708	0.000844	14.080605	0.019966	0.321728	0.111154	2.666907	2.666907
33709	0.000837	15.550569	-0.050830	0.278108	0.337711	3.831774	2.739662
33710	0.000610	15.284926	-0.161243	0.120775	0.497035	3.698490	0.805729
33711	0.000530	16.315661	0.172716	0.065682	0.467752	2.236469	1.109346

MultiFold introduction – Model

- https://energyflow.network/docs/archs/

Compilation Options

- **loss**= `'categorical_crossentropy'` : *str*
  - The loss function to use for the model. See the Keras loss function docs for available loss functions.
- **optimizer**= `'adam'` : Keras optimizer or *str*
  - A Keras optimizer instance or a string referring to one (in which case the default arguments are used).
- **metrics**= `['accuracy']` : *list* of *str*
  - The Keras metrics to apply to the model.
- **compile_opts**= `{}` : *dict*
  - Dictionary of keyword arguments to be passed on to the `compile` method of the model. `loss`, `optimizer`, and `metrics` (see above) are included in this dictionary. All other values are the Keras defaults.

Output Options

- **output_dim**= `2` : *int*
  - The output dimension of the model.
- **output_act**= `'softmax'` : *str* or Keras activation
  - Activation function to apply to the output.

Callback Options

- **filepath**= `None` : *str*
  - The file path for where to save the model. If `None` then the model will not be saved.
- **save_while_training**= `True` : *bool*
  - Whether the model is saved during training (using the `ModelCheckpoint` callback) or only once training terminates. Only relevant if `filepath` is set.
- **save_weights_only**= `False` : *bool*
  - Whether only the weights of the model or the full model are saved. Only relevant if `filepath` is set.
- **modelcheck_opts**= `{'save_best_only':True, 'verbose':1}` : *dict*
  - Dictionary of keyword arguments to be passed on to the `ModelCheckpoint` callback, if it is present. `save_weights_only` (see above) is included in this dictionary. All other arguments are the Keras defaults.
- **patience**= `None` : *int*
  - The number of epochs with no improvement after which the training is stopped (using the `EarlyStopping` callback). If `None` then no early stopping is used.
- **earlystop_opts**= `{'restore_best_weights':True, 'verbose':1}` : *dict*
  - Dictionary of keyword arguments to be passed on to the `EarlyStopping` callback, if it is present. `patience` (see above) is included in this dictionary. All other arguments are the Keras defaults.

MultiFold introduction – Model

- https://energyflow.network/docs/archs/#dnn

Required DNN Hyperparameters

- **input_dim** : *int* =6
  - The number of inputs to the model.
- **dense_sizes** : {*tuple*, *list*} of *int*=[100,100,100]
  - The number of nodes in the dense layers of the model.

Default DNN Hyperparameters

- **acts**=`'relu'` : {*tuple*, *list*} of *str* or Keras activation
  - Activation functions(s) for the dense layers. A single string or activation layer will apply the same activation to all dense layers. Keras advanced activation layers are also accepted, either as strings (which use the default arguments) or as Keras `Layer` instances. If passing a single `Layer` instance, be aware that this layer will be used for all activations and may introduce weight sharing (such as with `PReLU`); it is recommended in this case to pass as many activations as there are layers in the model.See the Keras activations docs for more detail.
- **k_inits**=`'he_uniform'` : {*tuple*, *list*} of *str* or Keras initializer
  - Kernel initializers for the dense layers. A single string will apply the same initializer to all layers. See the Keras initializer docs for more detail.
- **dropouts**=`0` : {*tuple*, *list*} of *float*
  - Dropout rates for the dense layers. A single float will apply the same dropout rate to all layers. See the Keras Dropout layer for more detail.
- **l2_regs**=`0` : {*tuple*, *list*} of *float*

Datasets: Run 12 200 GeV pp JP2 triggered

- PYTHIA and embedding: /gpfs01/star/pwg/elayavalli/ppRun12Embpicos, same as what Raghav and Isaac used
- Data: /gpfs01/star/pwg/elayavalli/ppRun12Datapicos
- PYTHIA cuts

Event	Track	Tower	Jet
events with jets having pT > 2 times the upper value of the pT-hard-bin dropped			anti-kT, R=0.4
	0.2 < p_T < 30 GeV	0.2 < E_T < 30 GeV	p_T > 5 GeV
	|eta| < 1	|eta| < 1	|eta| < 0.6

- Embedding and data cuts

Event	Track	Tower	Jet
events with jets having pT > 2 times the upper value of the pT-hard-bin, or with tracks/towers p_T / E_T > 30 GeV dropped			anti-kT, R=0.4
	0.2 < p_T < 30 GeV	0.2 < E_T < 30 GeV	p_T > 15 GeV
	|eta| < 1	|eta| < 1	|eta| < 0.6
	Primary		M > 1 GeV
JP2 triggered	DCA < 1 cm		neutral pT fraction < 0.9
|vz| < 30 cm	N_hits,fit >= 20		
	N_hits,frac >= 0.52		

Datasets: Run 12 200 GeV pp JP2 triggered

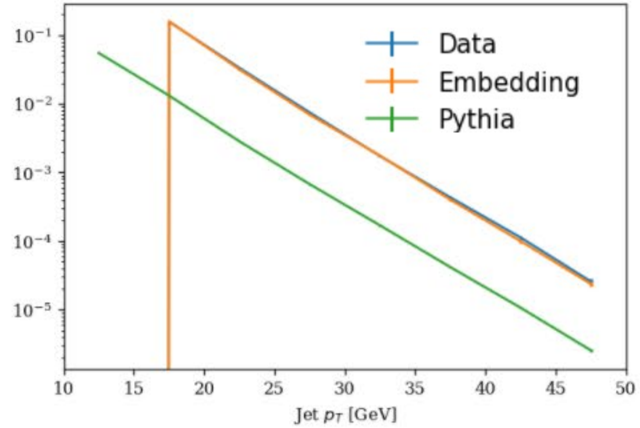
- Data and embedding agree nicely (fake fraction is small)

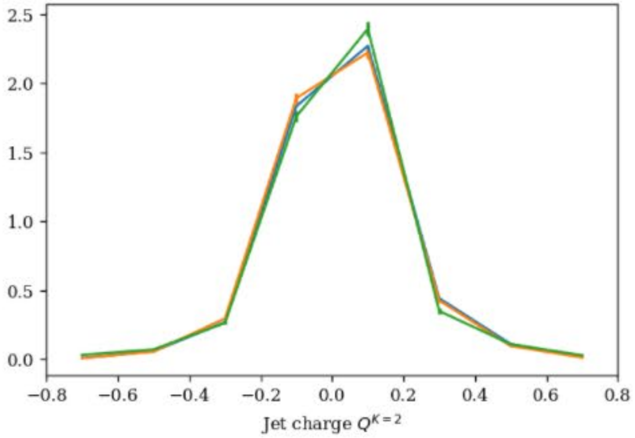
Different from the main slides, the substructure distributions here are not binned in pT

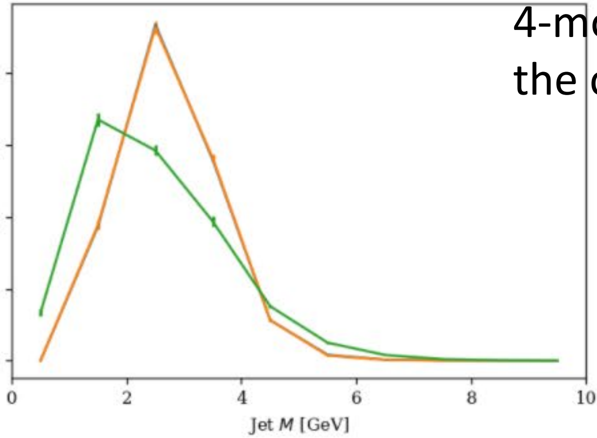
$$Q_J = \frac{1}{(p_{\mathrm{T}\,J})^{\kappa}} \sum_{i \in Tracks} q_i \times (p_{\mathrm{T,i}})^{\kappa}$$

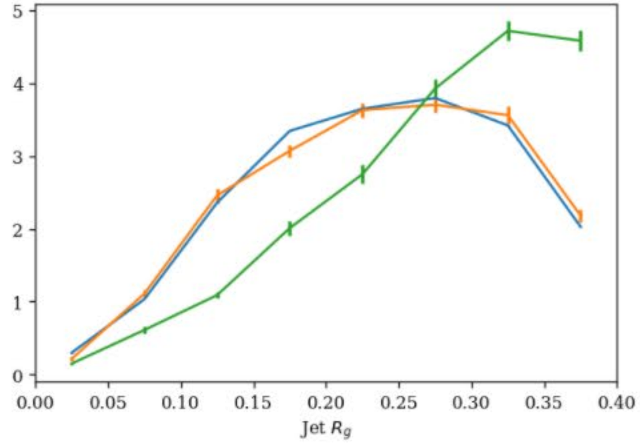
$$M = \left| \sum_{i \in \mathrm{jet}} p_i \right| = \sqrt{E^2 - \mathbf{p}^2};$$

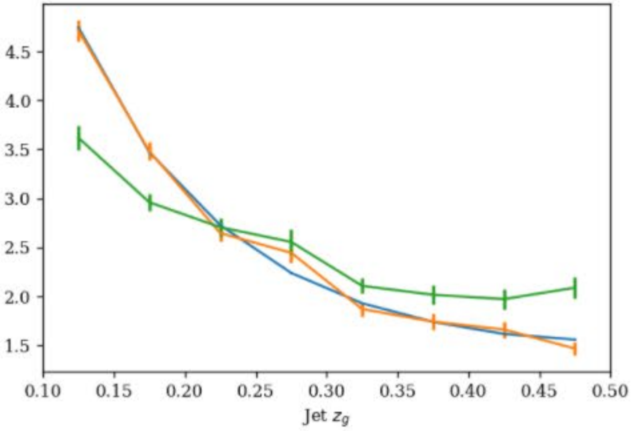
4-momentum of the constituent i

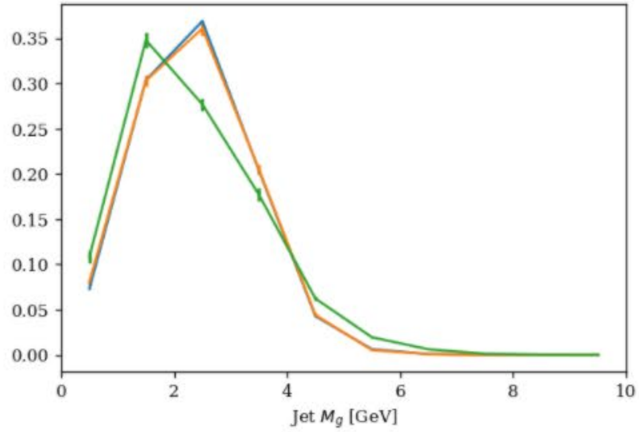








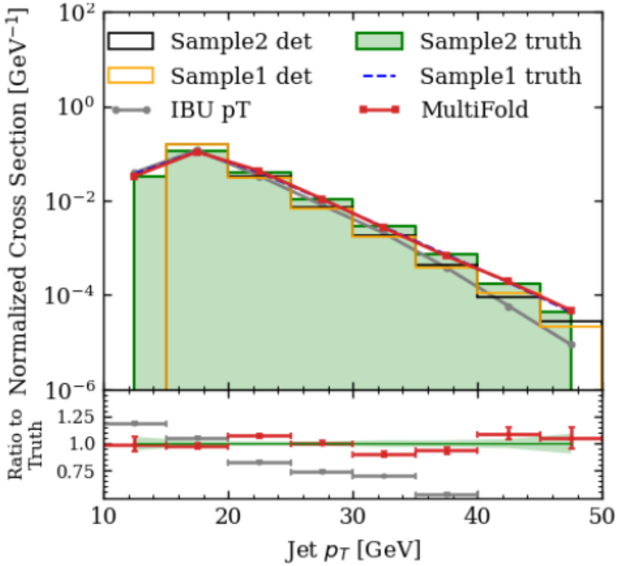


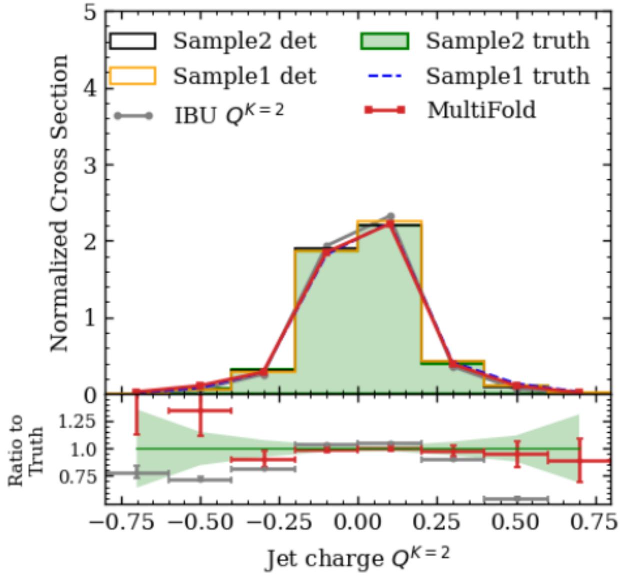


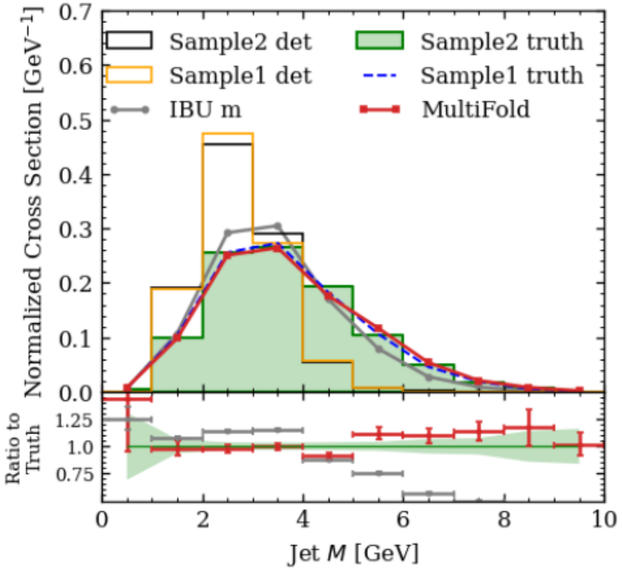
MultiFold closure test

➢ Closure test works!

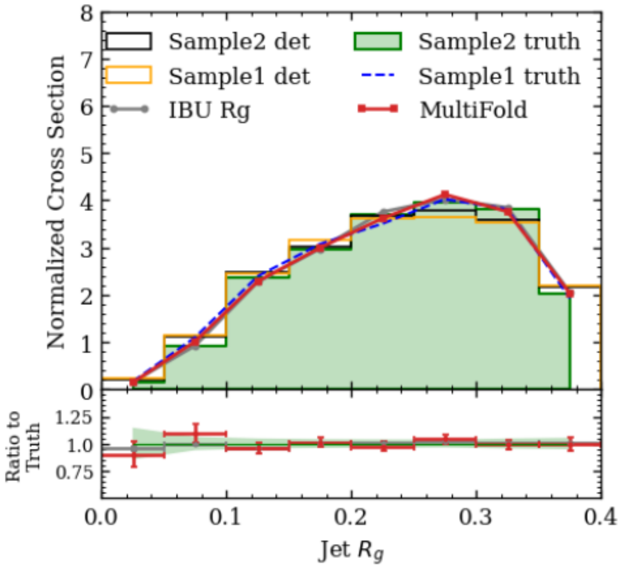
- Pythia and embedding samples are matched jets only

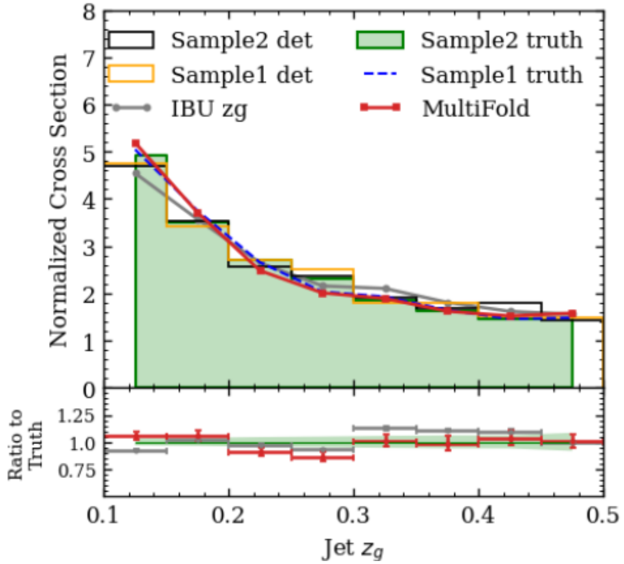


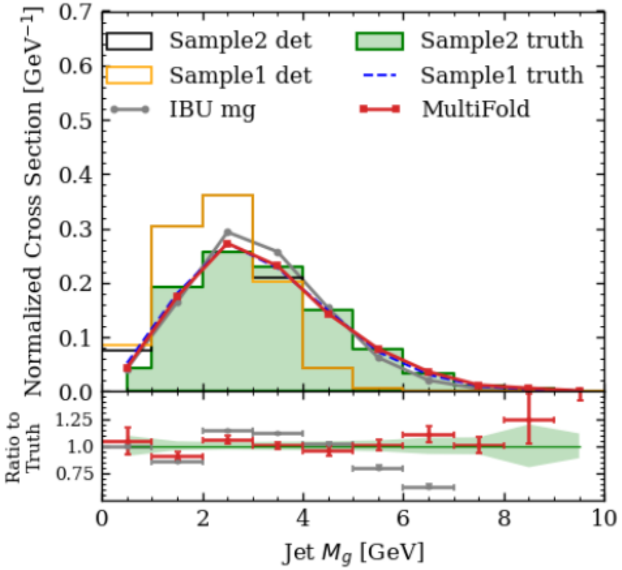




Seed = 1



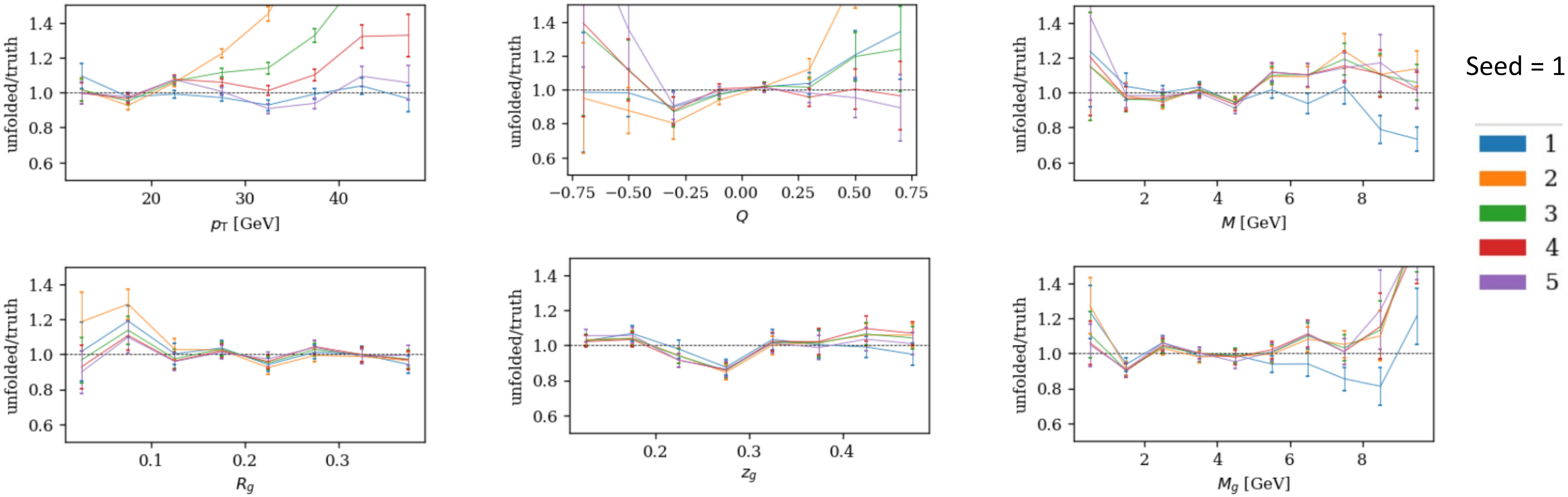




itnum = 4
patience1 = 50
patience2 = 50
valnum = 0.2
batch_size = 10000

MultiFold closure test

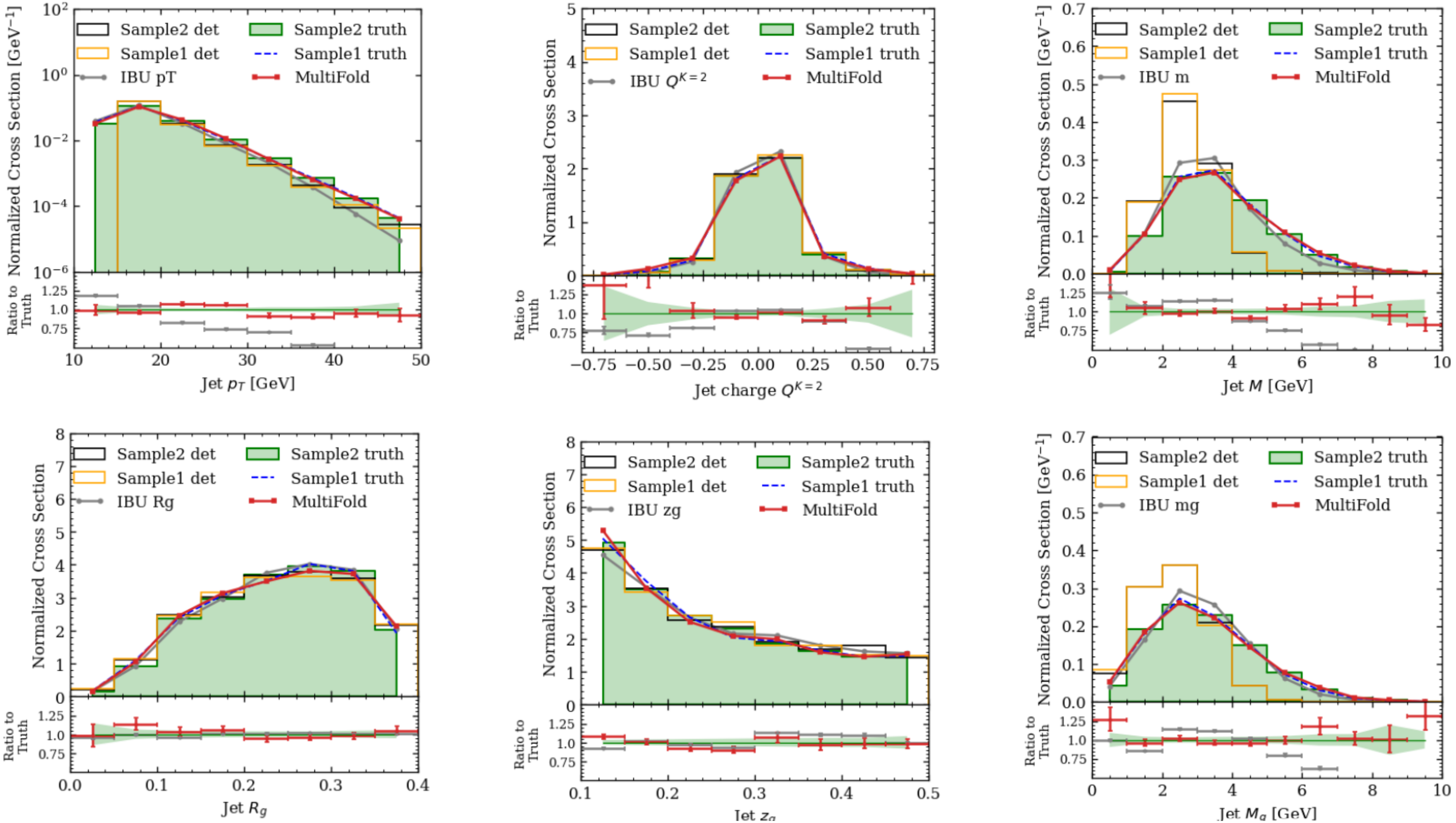
➢ How many iterations to choose ?



MultiFold closure test

- Closure test works!

- Pythia and embedding samples are matched jets only



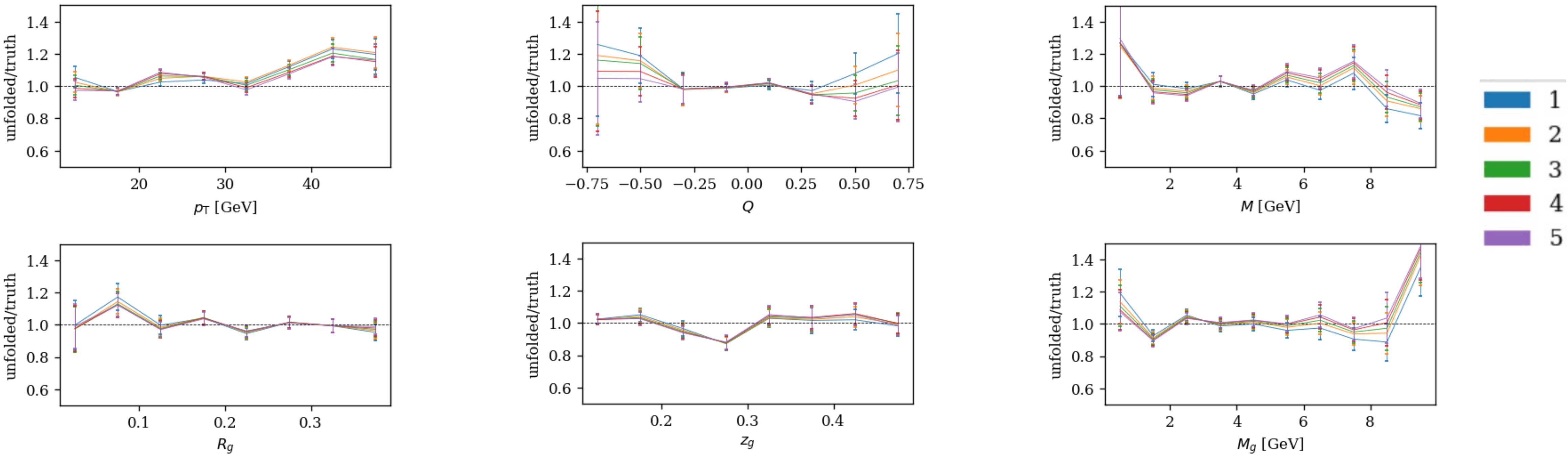
Seed = 0

itnum = 4
patience1 = 50
patience2 = 50
valnum = 0.2
batch_size = 10000

MultiFold closure test

- After averaging over 100 seeds, unfolded spectrum has a smaller dependence on the number of iteration, and closure is also better.

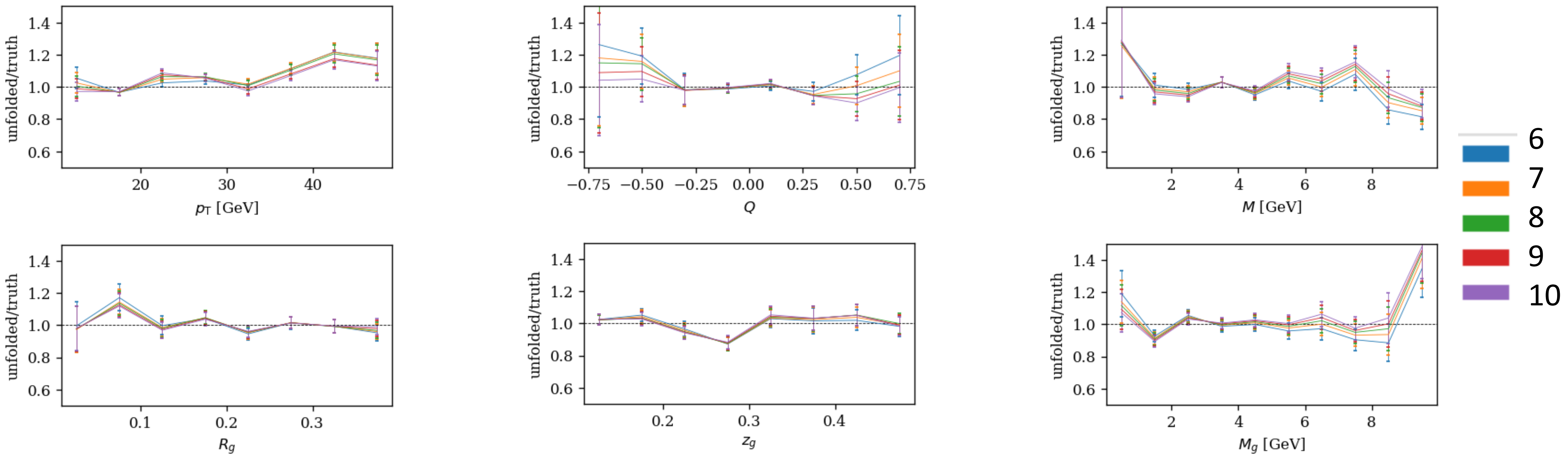
Average over 100 seeds



MultiFold closure test

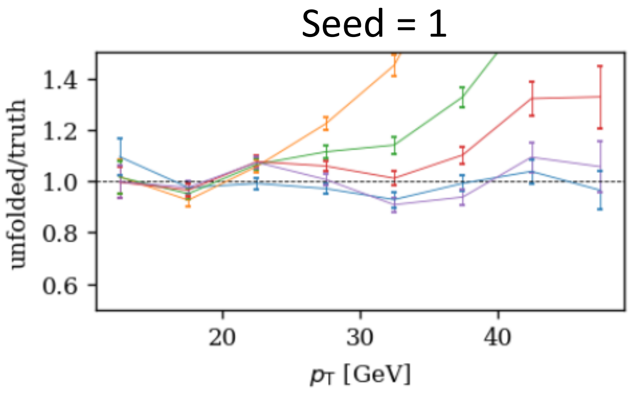
- After averaging over 100 seeds, unfolded spectrum has a smaller dependence on the number of iteration, and closure is also better.

Average over 100 seeds



- Anything between 2 to 10 iterations gives reasonable closure. We will see that in data, **variation due to different number of iterations is much smaller than the statistical uncertainty.**

MultiFold closure test

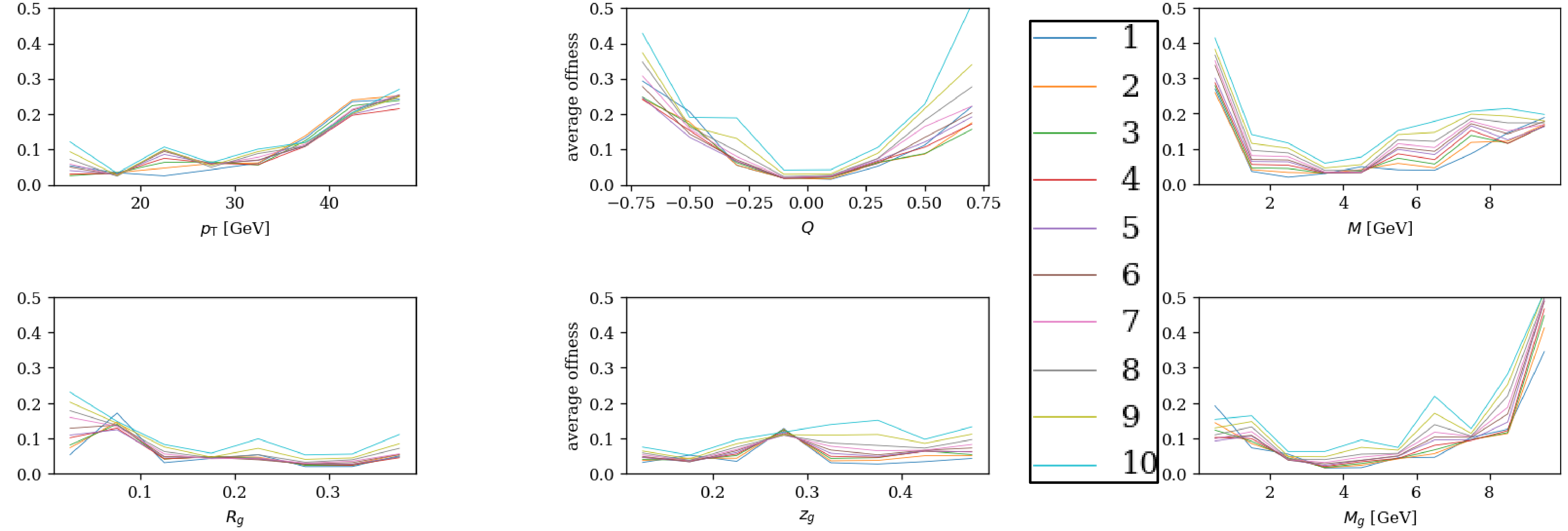


In the last pT bin, unfolded spectrum is “off” by ~30% in the 4th iteration, for this seed. Check the average “offness” over 100 seeds.

MultiFold closure test

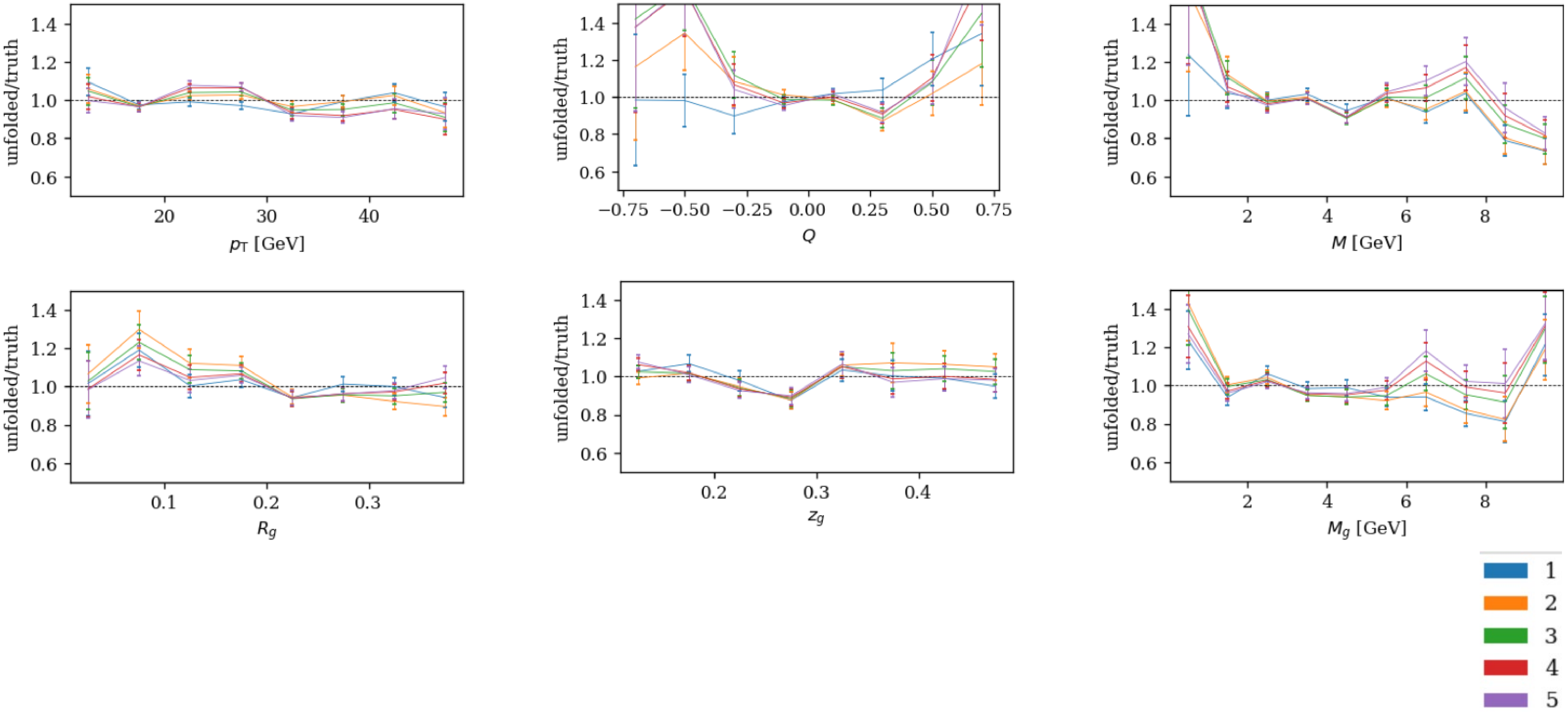
- y axis = avg{abs[(normalized multifolded spectrum for a given seed/ normalized truth spectrum) – 1]}

Average over 100 seeds



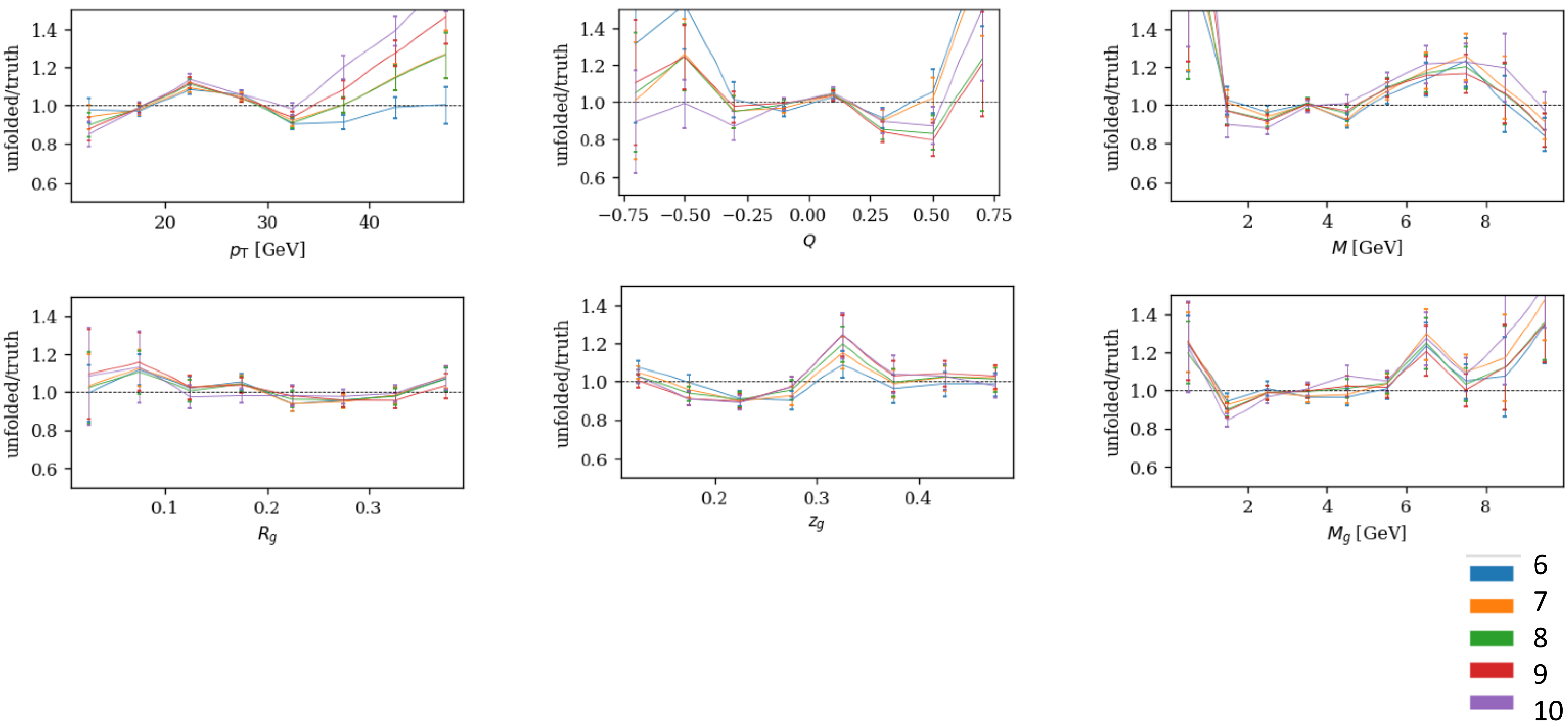
MultiFold closure test

Seed = 0



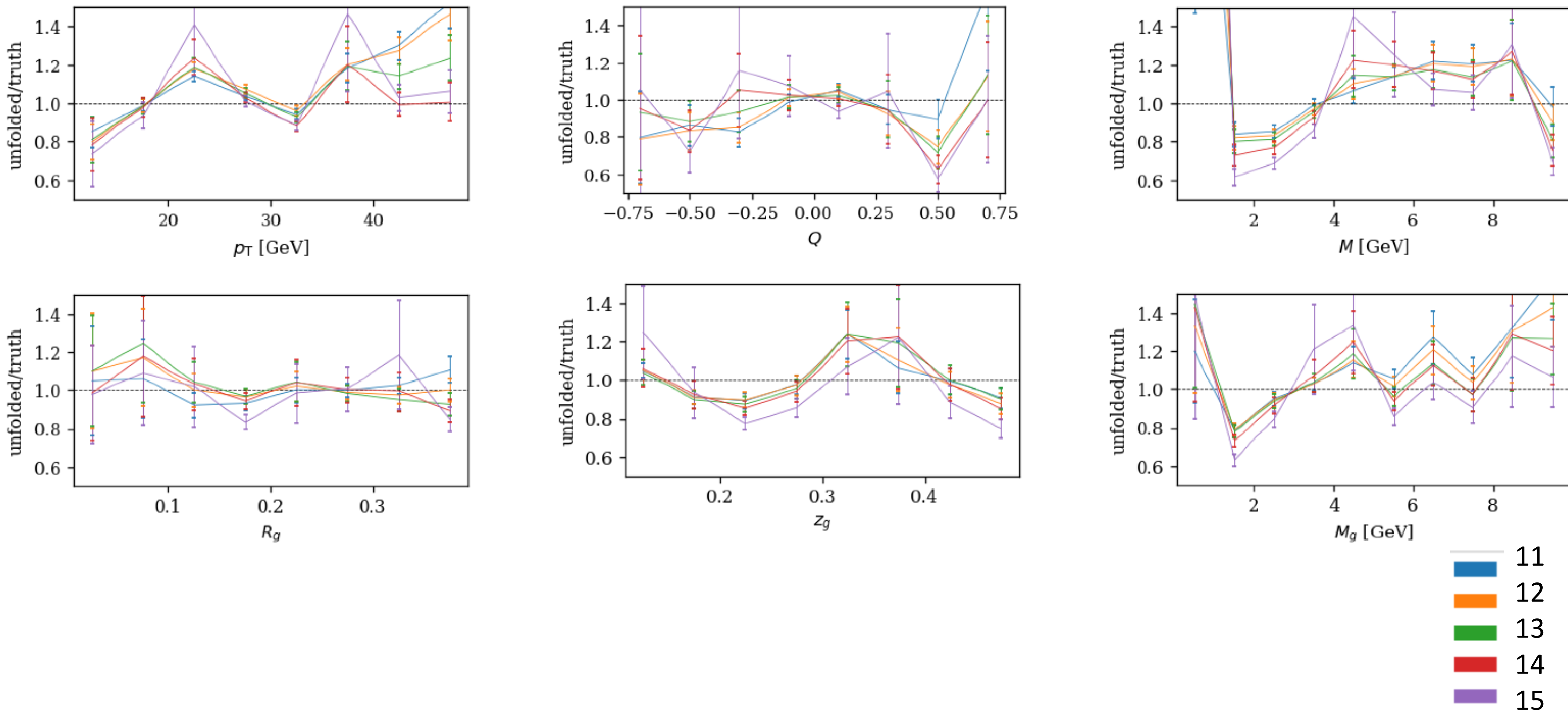
MultiFold closure test

Seed = 0



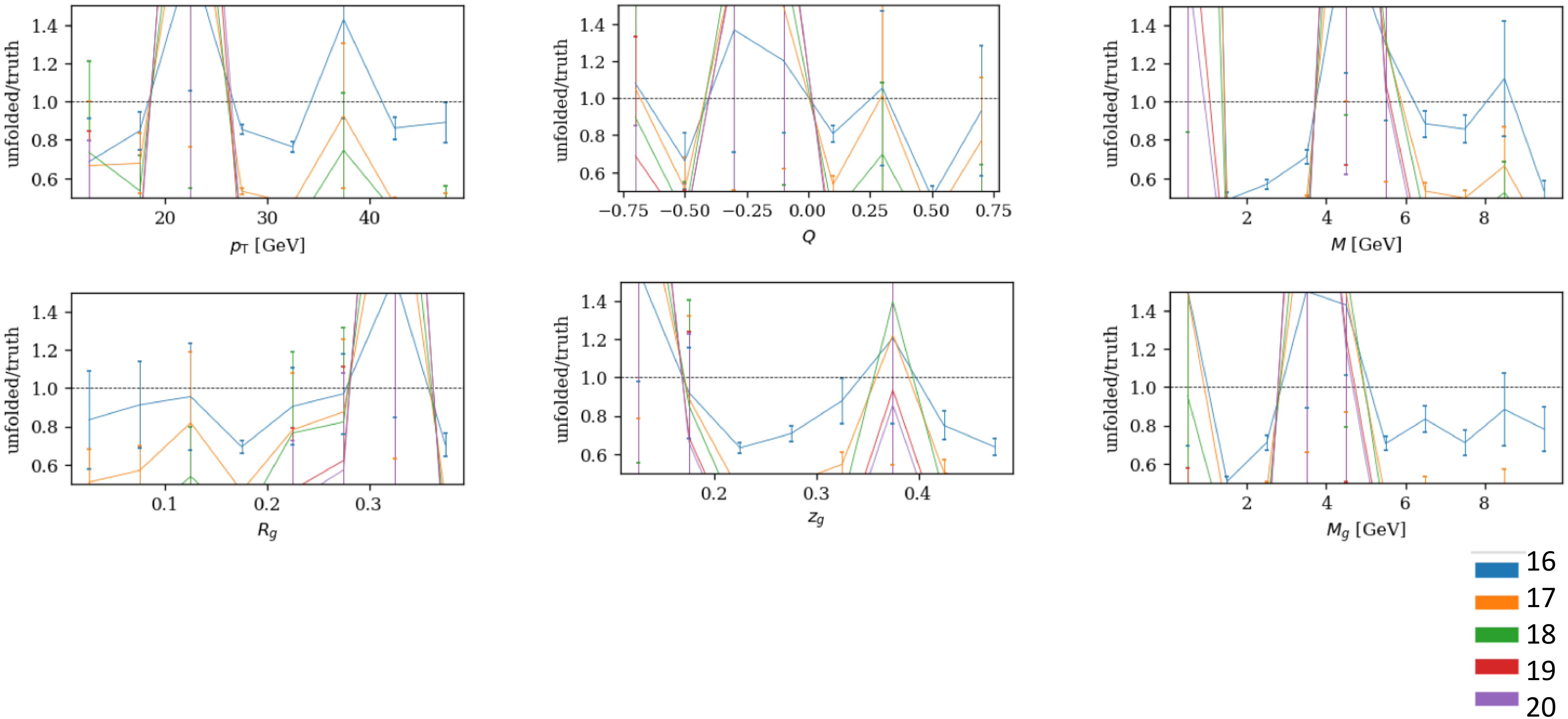
MultiFold closure test

Seed = 0

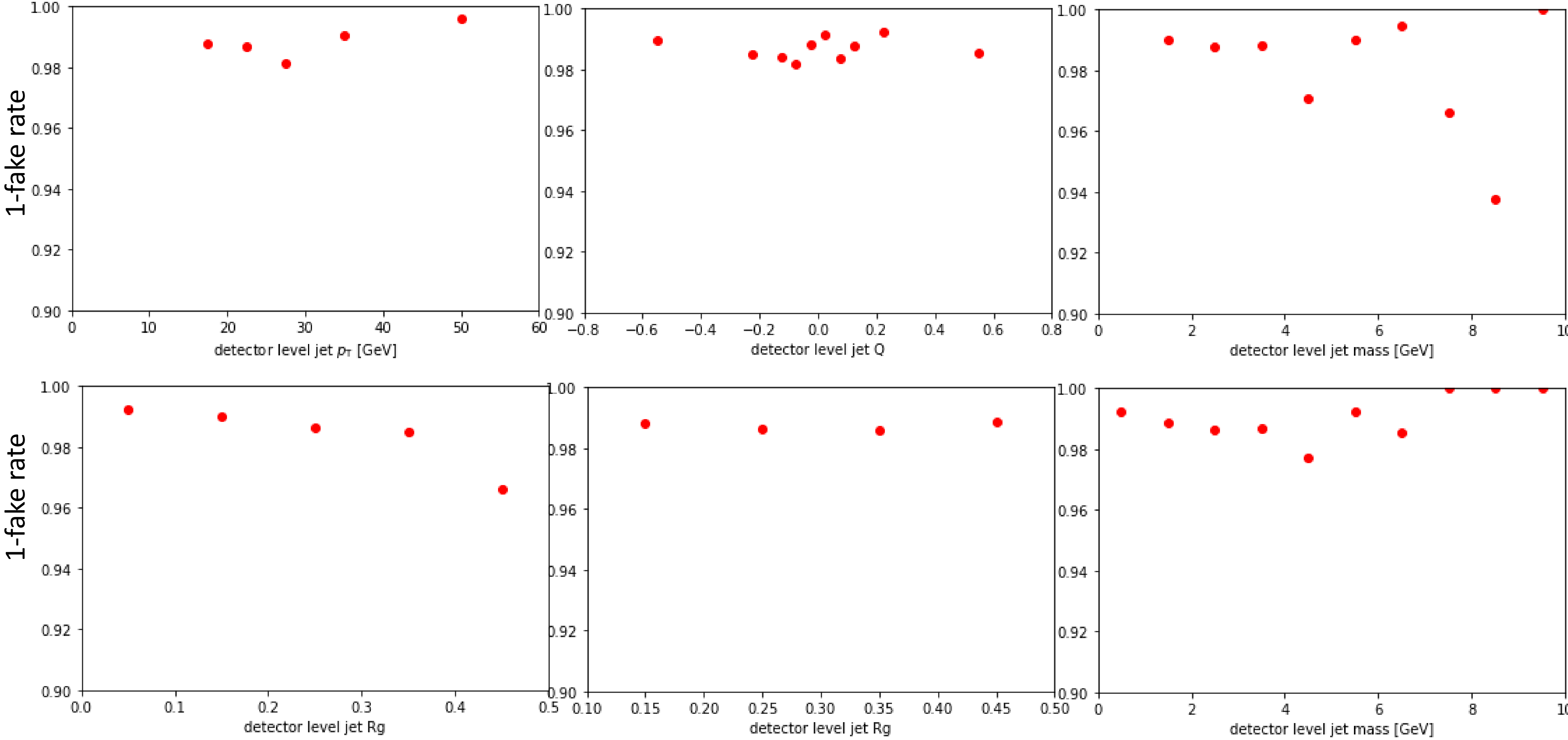


MultiFold closure test

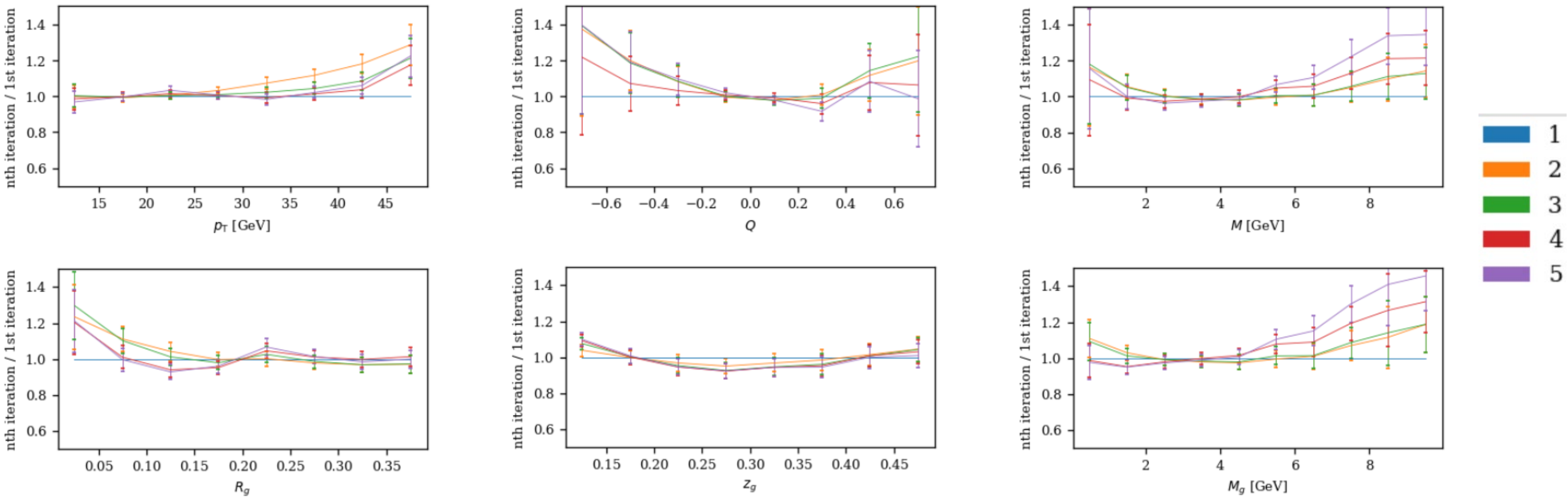
Seed = 0



MultiFold on data



MultiFold on data
How does the number of iterations affect the unfolding result?



- Done with a given seed to show the extent of variation only due to the number of iterations

Full spectrum comparison - pT

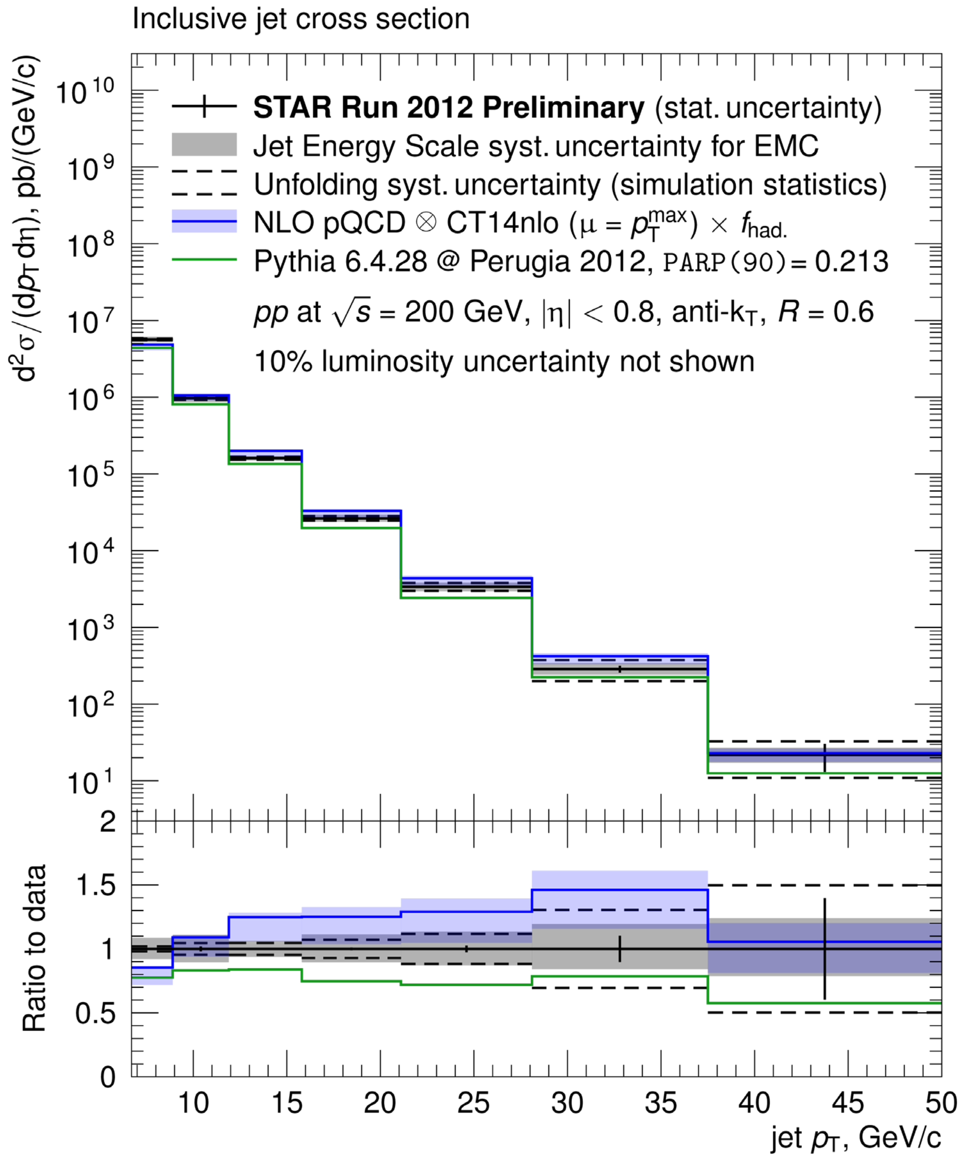
To compare with the results from

https://drupal.star.bnl.gov/STAR/system/files/preliminary_release_run12pp200jets.pdf

We use:

Jet p_T corrected for UE, GeV/c	Differential cross section $\frac{d^2\sigma}{dp_T\, d\eta}$, pb
6.70 – 8.90	$5.7 \cdot 10^6$
8.90 – 11.9	$9.7 \cdot 10^5$
11.9 – 15.8	$1.6 \cdot 10^5$
15.8 – 21.1	$2.6 \cdot 10^4$
21.1 – 28.1	$3.4 \cdot 10^3$
28.1 – 37.5	$2.9 \cdot 10^2$
37.5 – 50.0	$2.2 \cdot 10^1$

Figure 5: Numerical values for differential Inclusive Jet cross section for proton-proton collisions at $\sqrt{s} = 200$ GeV.



Full spectrum comparison - pT

To compare with the results from

https://drupal.star.bnl.gov/STAR/system/files/preliminary_release_run12pp200jets.pdf *

We use:

Method for normalization for the nominal value:

Let y2 = MultiFolded pT distribution (binned as in *),
eff= Efficiency (binned as in *),

Solve for n and c from

n*(y2[4]/eff[4])=3400

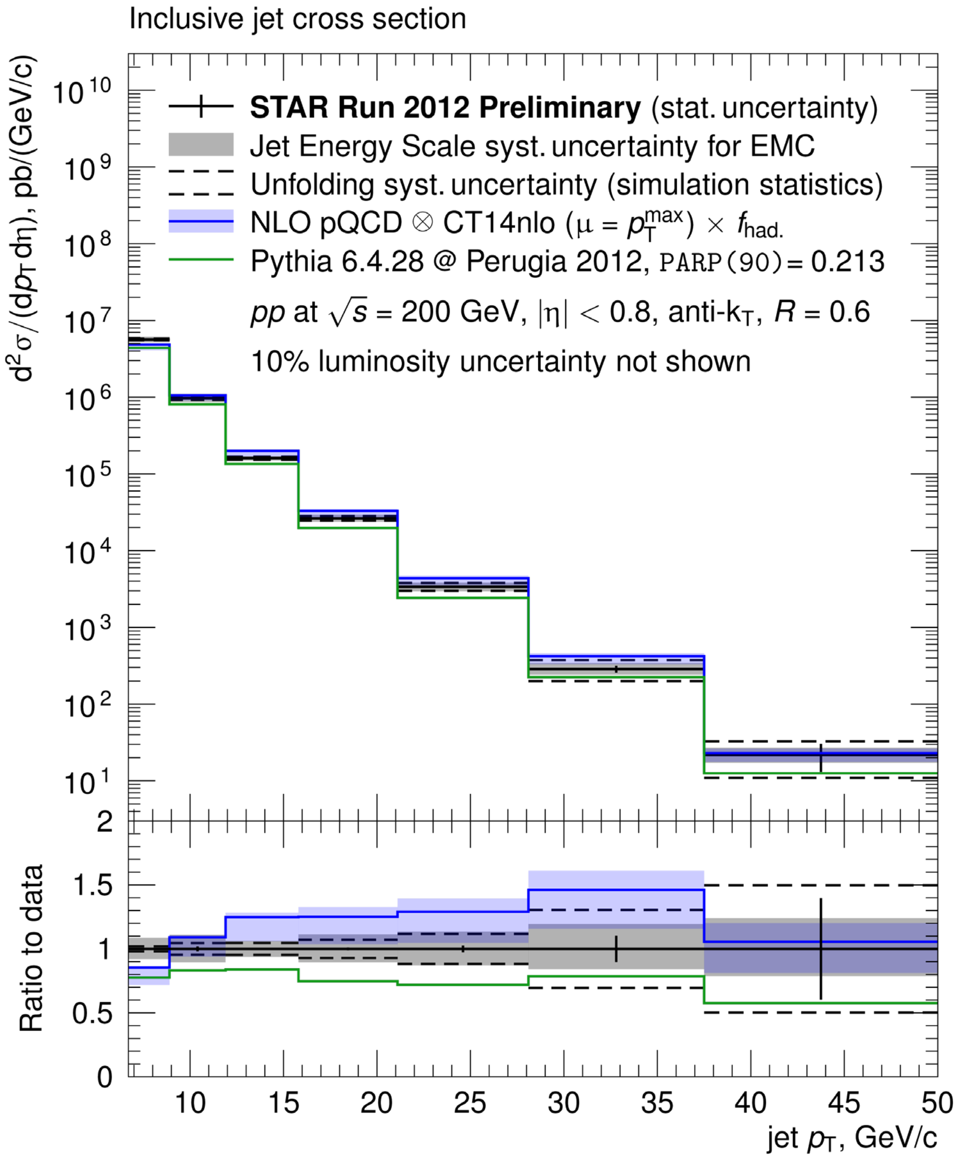
Then

Nominal = n*y2/eff

Note: Pythia pT distribution for misses not used.

Jet p_T corrected for UE, GeV/c	Differential cross section $\frac{d^2\sigma}{dp_T\,d\eta}$, pb
6.70 – 8.90	$5.7 \cdot 10^6$
8.90 – 11.9	$9.7 \cdot 10^5$
11.9 – 15.8	$1.6 \cdot 10^5$
15.8 – 21.1	$2.6 \cdot 10^4$
21.1 – 28.1	$3.4 \cdot 10^3$
28.1 – 37.5	$2.9 \cdot 10^2$
37.5 – 50.0	$2.2 \cdot 10^1$

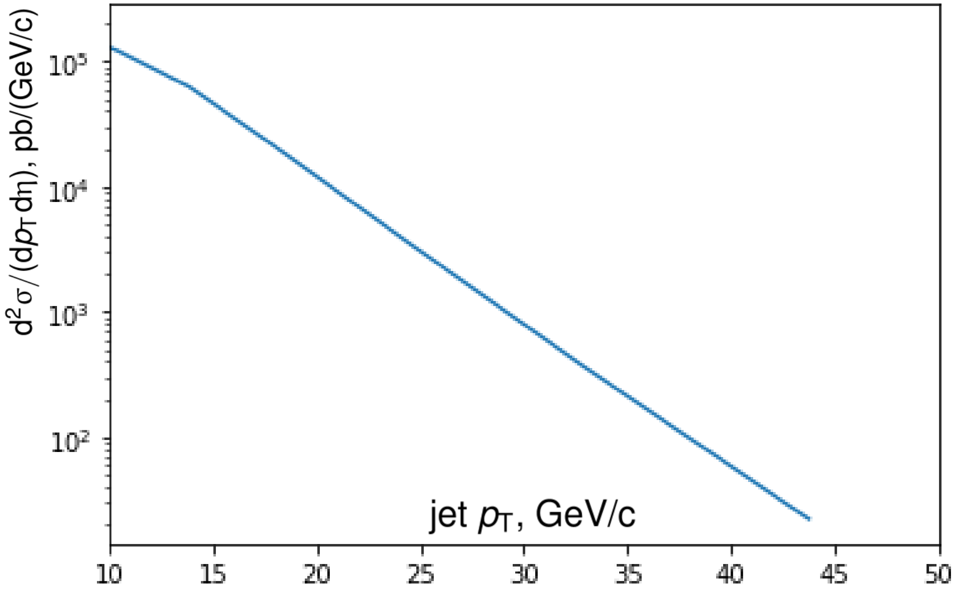
Figure 5: Numerical values for differential Inclusive Jet cross section for proton-proton collisions at $\sqrt{s} = 200$ GeV.

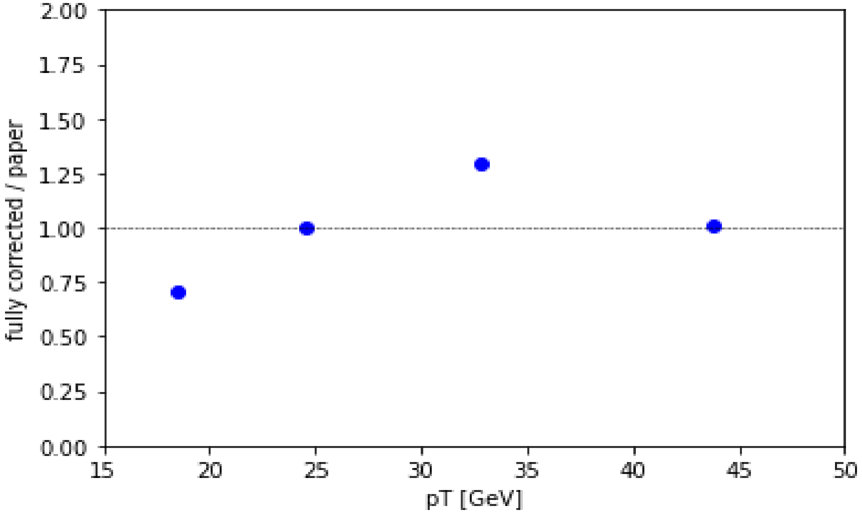


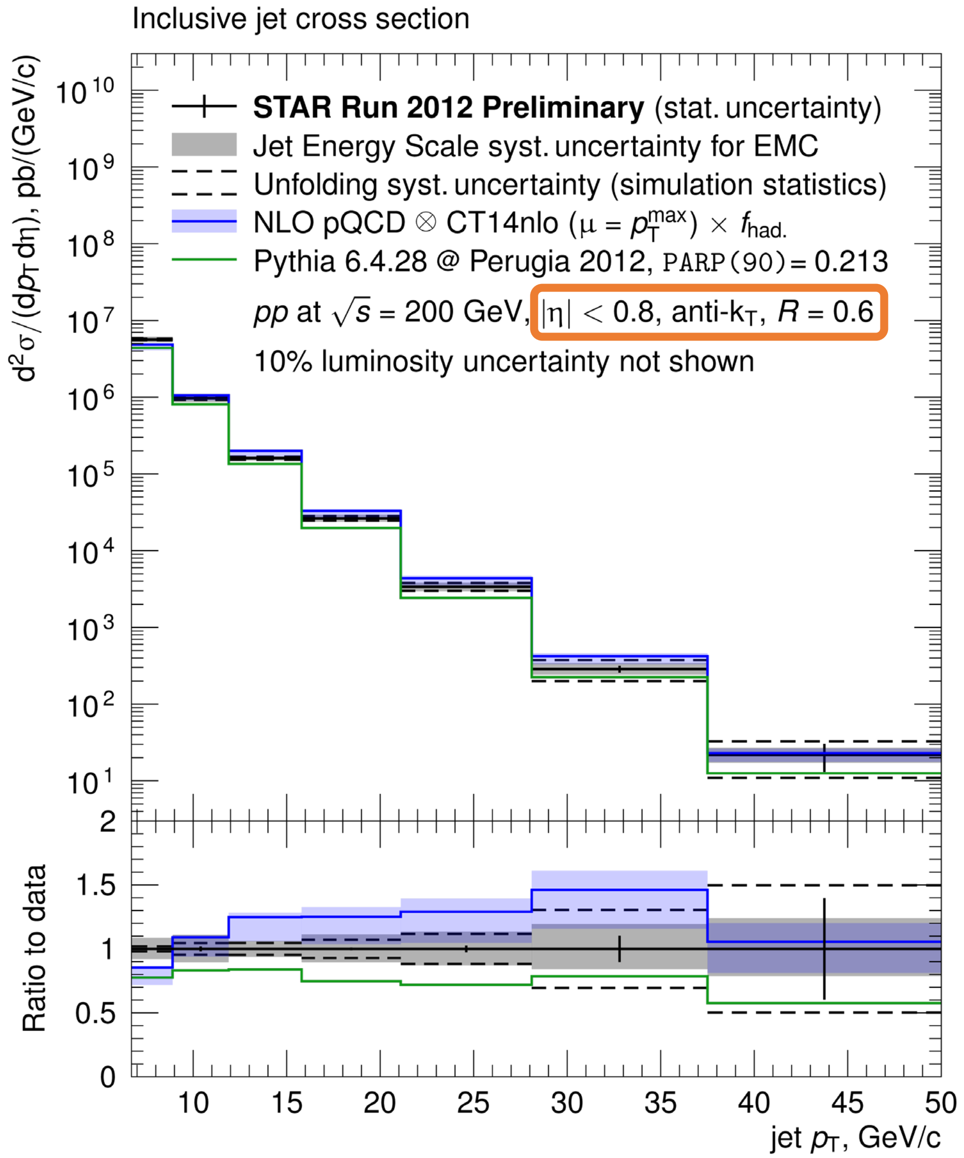
Full spectrum comparison - pT

To compare with the results from

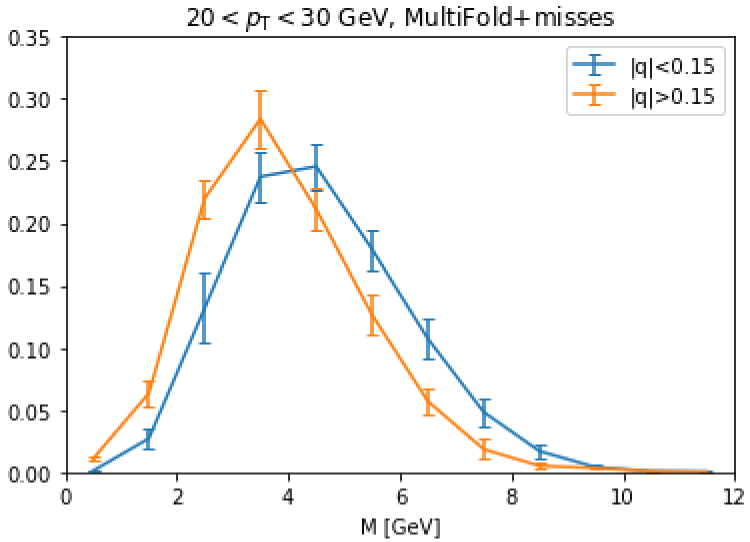
https://drupal.star.bnl.gov/STAR/system/files/preliminary_release_run12pp200jets.pdf *

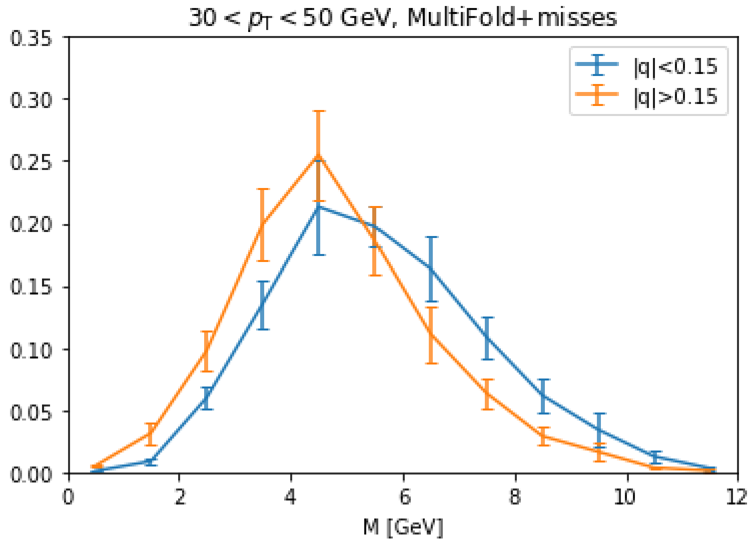


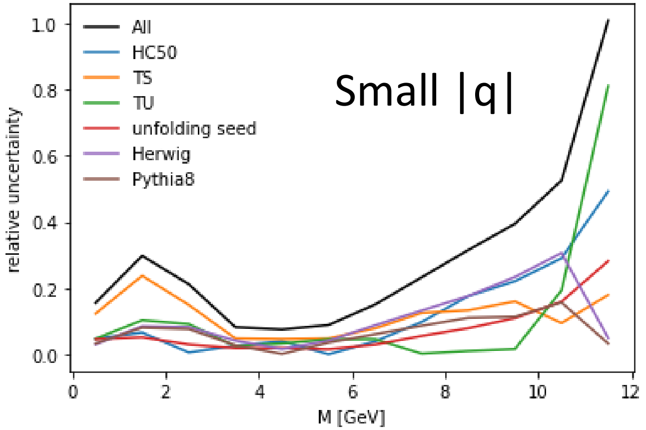


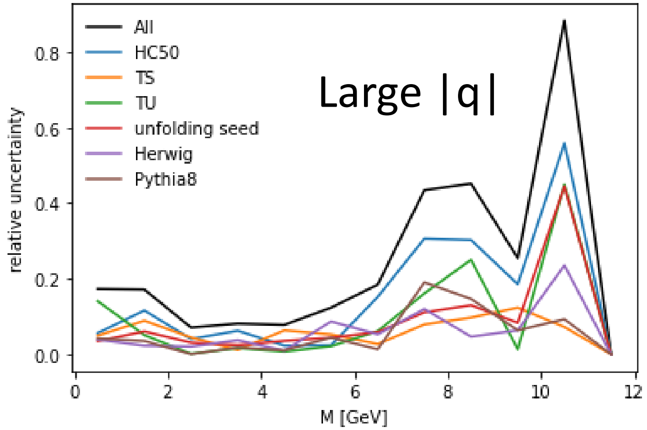


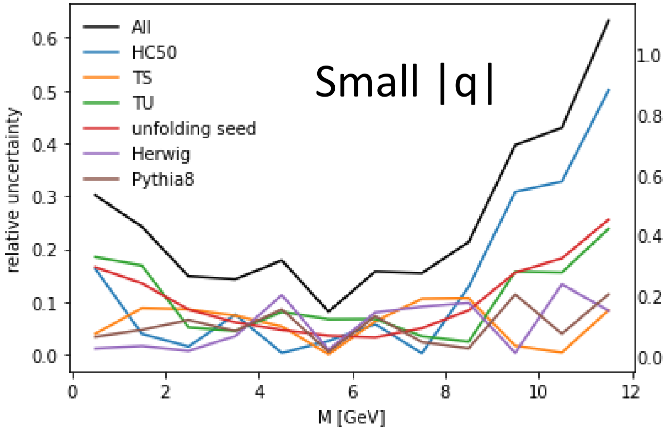
Mass vs charge

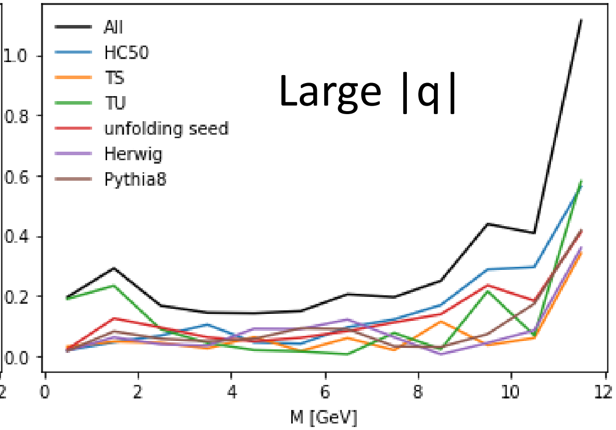












Also ran KS tests, but need to figure out what normalization option to use. Either way values ~ 0.