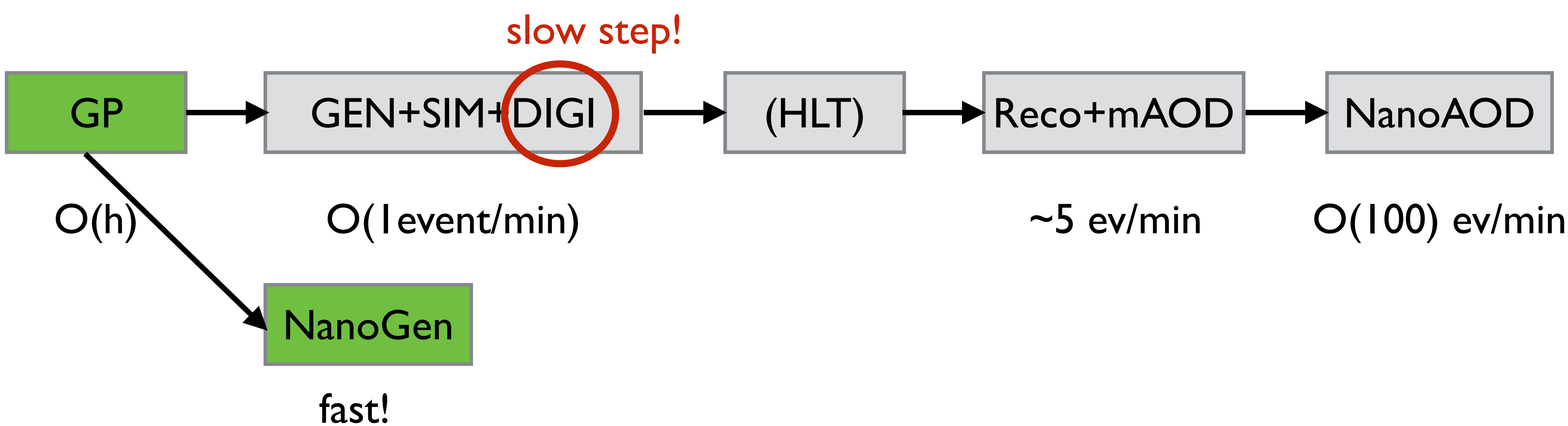
Event generation for EFT samples

Second LPC EFT Workshop @ Notre Dame
April 22, 2024

Daniel Spitzbart (Boston U)

Generation qorkflow at a glance

- Not too many differences for EFT event generation compared to the "standard" CMS workflow
  - BUT: we definitely want to add EFT information: weights and coefficents
- Starting from a MG gridpack (see next slides):
  /eos/uscms/store/user/dspitzba/TT01j_tutorial_slc7_amd64_gcc700_CMSSW_10_6_19_tarball.tar.xz
- cmsDriver commands for full example chain can be found on PdmV twikis: e.g. UL18
- Interest of time: Go to NanoGEN directly, but all instructions work just as well for a "full" NanoAOD configuration



Telling madgraph what to do

- We want to generate a “gridpack” containing all the information about our favorite process
- A gridpack contains all executable Madgraph_aMC@NLO code necessary for event generation
- To generate this gridpack we need a set of files or cards:
    - The physics model, provided as UFO (Universal FeynRules Output)
    - Proc card: specifies the process definition
    - Run card: specifies the run parameters
    - Reweight card: specifies each of the new physics scenarios that will be probed
    - Some more cards can be added like the restrict card or customize card

Process card

Process card

```
1    import model SMEFTsim_topU3l_MwScheme_UFO
2
3    define p = g u c d s b u~ c~ d~ s~ b~
4    define j = p
5    define l+ = e+ mu+ ta+
6    define l- = e- mu- ta-
7    define vl = ve vm vt
8    define vl~ = ve~ vm~ vt~
9    define had = u c d s u~ c~ d~ s~
10
11   generate    p p > t t~      @0  NPprop=0 SMHLOOP=0 NP=1
12   add process p p > t t~ j    @1  NPprop=0 SMHLOOP=0 NP=1
13   #add process p p > t t~ j j  @2  NPprop=0 SMHLOOP=0 NP=1
14
15   # in 2l decay:
16   #generate    p p > t t~, t > l+ vl b, t~ > l- vl~ b~      @0  NPprop=0 SMHLOOP=0 NP=1
17   #add process p p > t t~ j, t > l+ vl b, t~ > l- vl~ b~    @1  NPprop=0 SMHLOOP=0 NP=1
18
19   output TT01j_tutorial -nojpeg
```

Define particles:
all particles defined in line 3 will be considered as initial state particles

Define the process

Model specific flags are added

Generates a folder with all the code needed

Run card

Run card

```
#*********************************************************************
# Minimum and maximum pt's (for max, -1 means no cut)                *
#*********************************************************************
  10 = ptj        ! minimum pt for the jets
  0  = ptb        ! minimum pt for the b
  0  = pta        ! minimum pt for the photons
```

Minimum pt of objects is defined

```
#*********************************************************************
 5 = maxjetflavor    ! Maximum jet pdg code
#*********************************************************************
# Jet measure cuts                                                   *
#*********************************************************************
 10  = xqcut   ! minimum kt jet measure between partons
#*********************************************************************
```

maxjetflavor sets the flavor scheme

xqcut is the matching parameter

Reweight card

Reweight card

```
launch --rwgt_name=EFTrwgt1_ctGRe_1.0_ctGIm_0.0_ctWRe_0.0_ctWIm_0.0_ctBRe_0.0_ctBIm_0.0_cHtbRe_0.0_cHtbIm_0.0_cHt_0.0
set ctGRe 1.000000
set ctGIm 0.000000
set ctWRe 0.000000
set ctWIm 0.000000
set ctBRe 0.000000
set ctBIm 0.000000
set cHtbRe 0.000000
set cHtbIm 0.000000
set cHt 0.000000
```

All possible new physics scenarios are explored and weights are calculated corresponding to each scenario

```
launch --rwgt_name=EFTrwgt2_ctGRe_0.0_ctGIm_1.0_ctWRe_0.0_ctWIm_0.0_ctBRe_0.0_ctBIm_0.0_cHtbRe_0.0_cHtbIm_0.0_cHt_0.0
set ctGRe 0.000000
set ctGIm 1.000000
set ctWRe 0.000000
set ctWIm 0.000000
set ctBRe 0.000000
set ctBIm 0.000000
set cHtbRe 0.000000
set cHtbIm 0.000000
set cHt 0.000000
```

Generating a gridpack

- Follow instructions on github repository
- Commands have been tested on the LPC cluster (cmslpc-el8) and CERN lxplus (lxplus8)
  - You might need to run some parts in the right singularity containers → be careful
- We use central CMS tools (genproductions) that are complemented with not-yet-central EFT specific tools (EFTfit)
  - Syntax and naming should be followed with care!

Creating an EFT NanoGEN sample

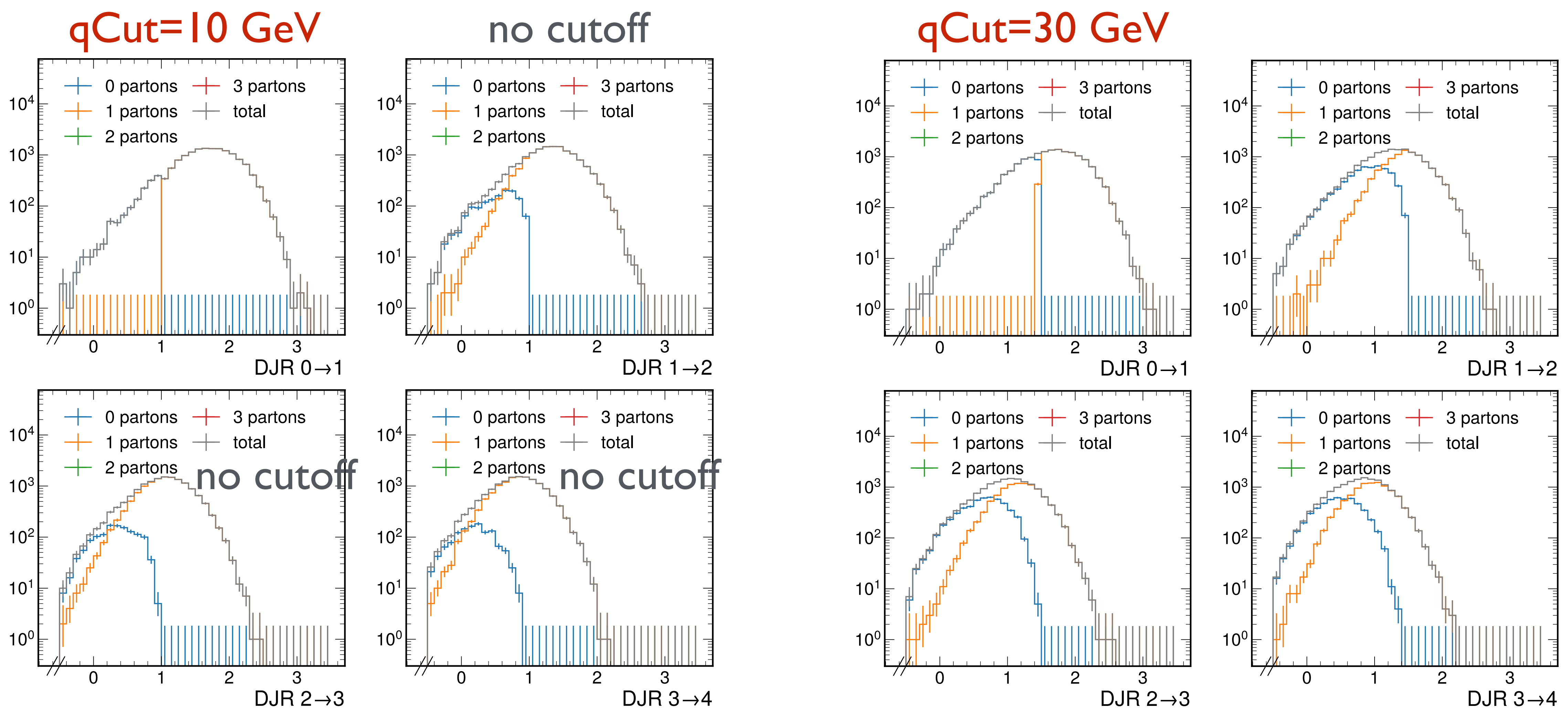
- Reminder: produced gridpack with multiple points in EFT parameter space
- Gridpack + Pythia fragment + cmsDriver commands → CMS sample
- How to keep the weights + coordinates in EFT space?
  - Keep the weights + names: use NamedWeights in NanoAOD / NanoGEN
  - Use code that extracts the polynomial coefficients
- For keeping weights we need to know the name which is set in the reweight_card, suffix depends on the reweighting method employed ("change mode …"). Add weights to the NanoGEN configuration

```
change rwgt_dir rwgt

launch --rwgt_name=dummy

launch --rwgt_name=EFTrwgt0_ctGRe_0.0_ctGIm_0.0_ctWRe_0.0_ctWIm_0.0_ctBRe_0.0_ctBIm_0.0_cHtbRe_0.0_cHtbIm_0.0_cHt_0.0
set ctGRe 0.000000
set ctGIm 0.000000
set ctWRe 0.000000
```

- Using EFTGenReader package to extract coefficients, expects a certain naming scheme: "EFTrwgtN_{coeff}_{value}_…"

Interlude: obtaining qcut values

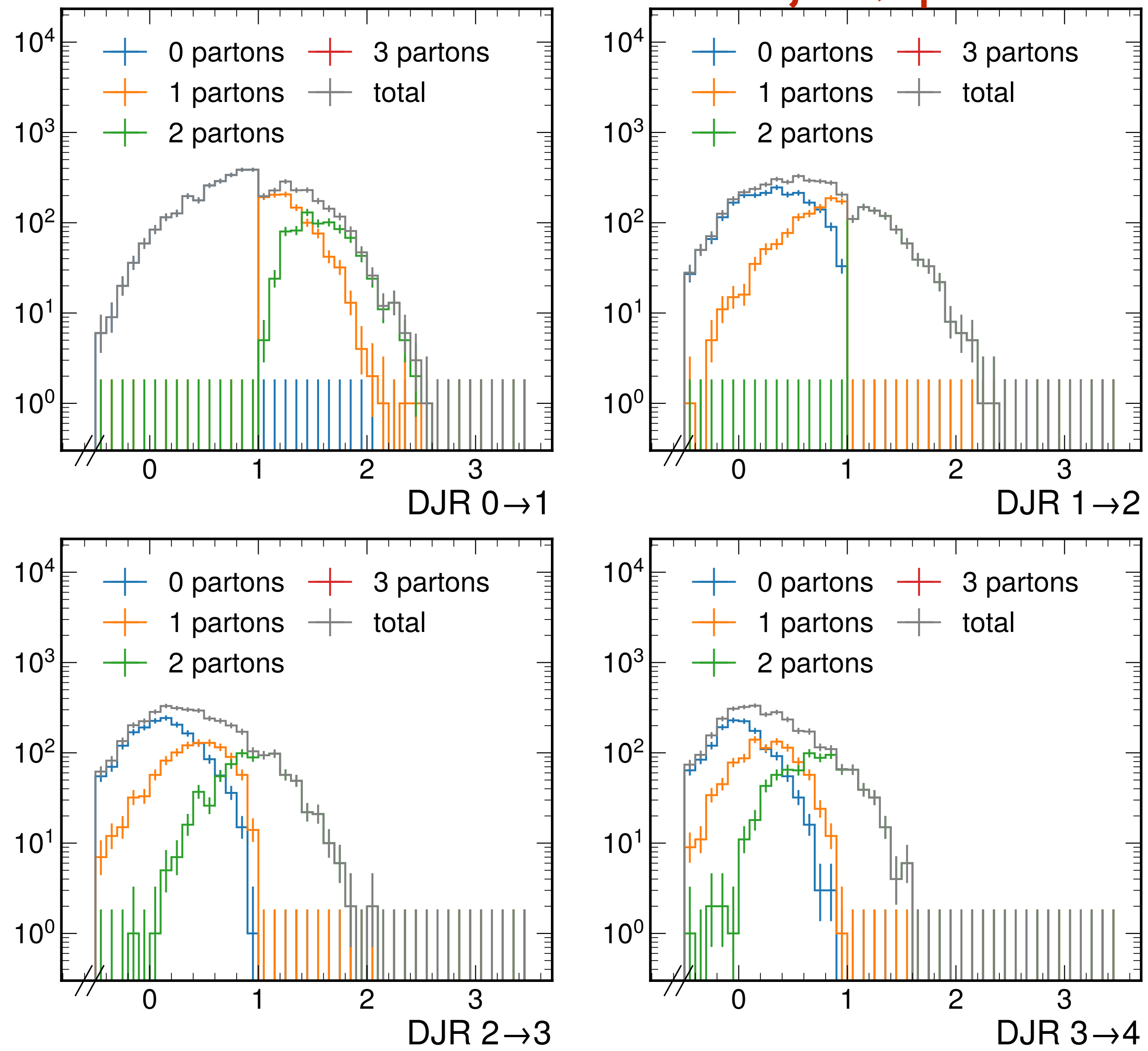
- Important topic for any sample with additional partons at ME level, e.g. W+jets, tt+jets, …
- Have to ensure that transition between ME (MadGraph) and PS (pythia) is smooth
  - Differential jet rate (DJR) distribution is a good measure for that
- DJR corresponds to the k_T separation for a given jet multiplicity
  - Example: For 2 jets with Δk_T =20 GeV we get DJR(1➝2)=20 GeV



Bad qCut choices

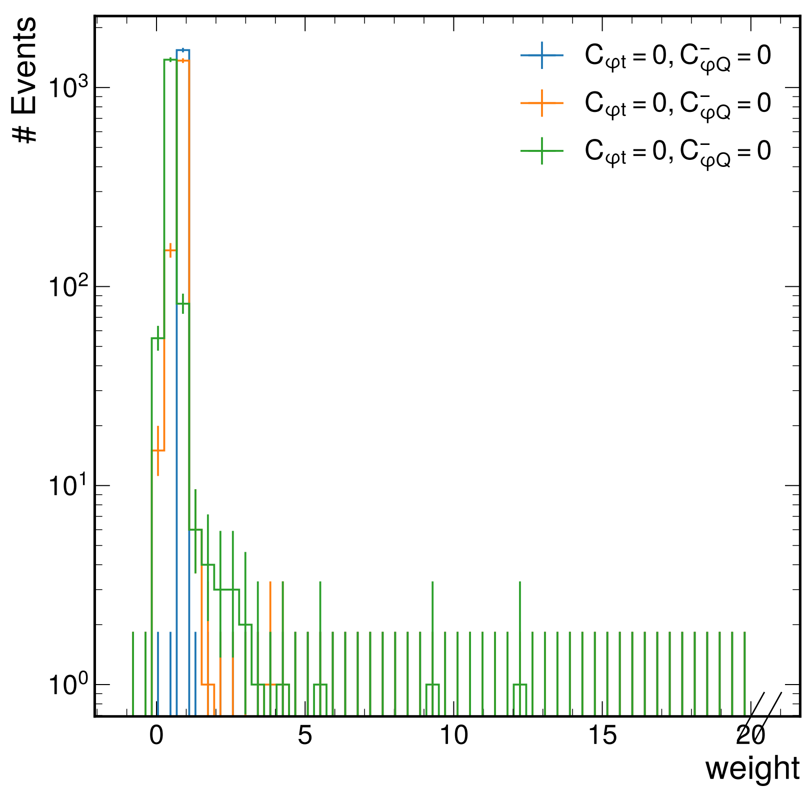
- General rule of thumb: Don't go too high in qCut, otherwise you'll mainly keep events from the parton shower
  - Then what's the point of adding extra partons in the ME
  - Can also become very inefficient

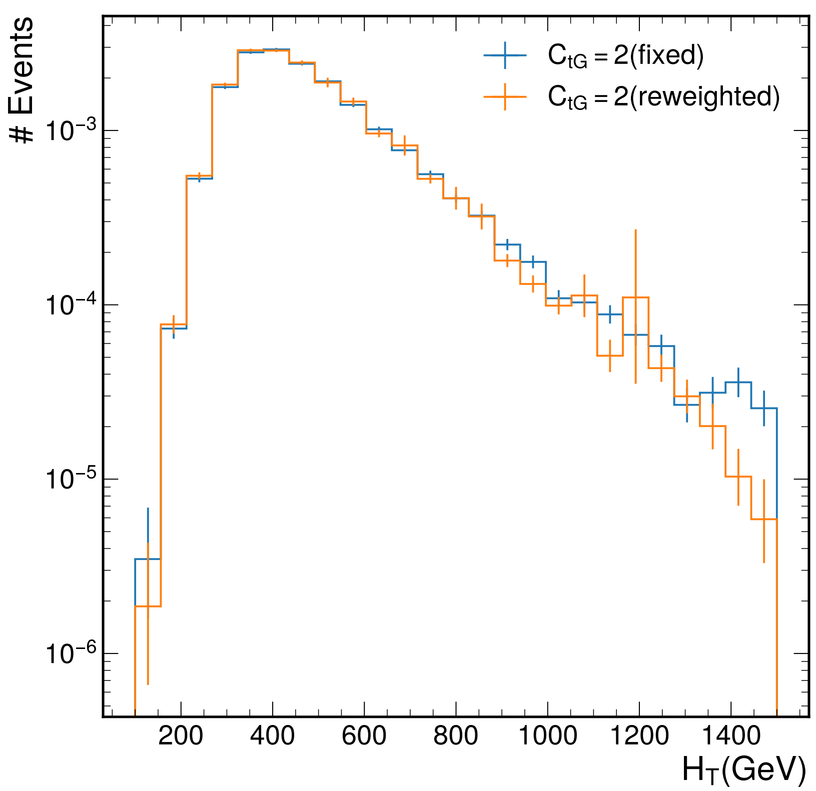
W+jets, qCut=10 GeV



Validating a sample

- Weights and coefficient distributions
    - Are there big tails in the distributions of weights?
- Comparing with a fixed-point sample
    - Do we actually reproduce distributions with reasonable precision?
    - Create a gridpack + sample at some interesting point in EFT space, using the customize card



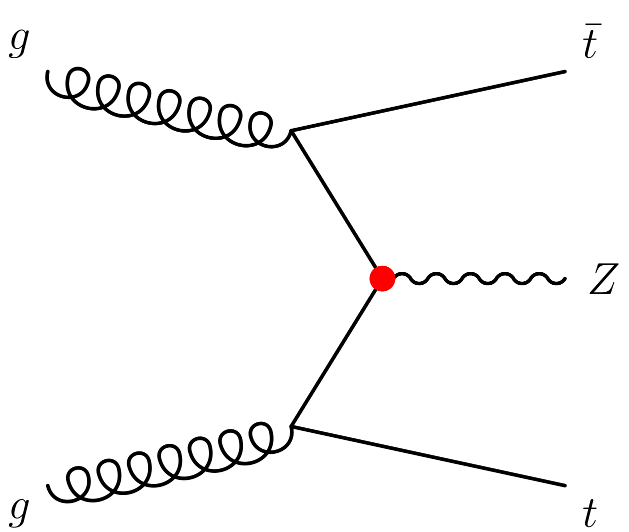


BACKUP

Event generation in a nutshell

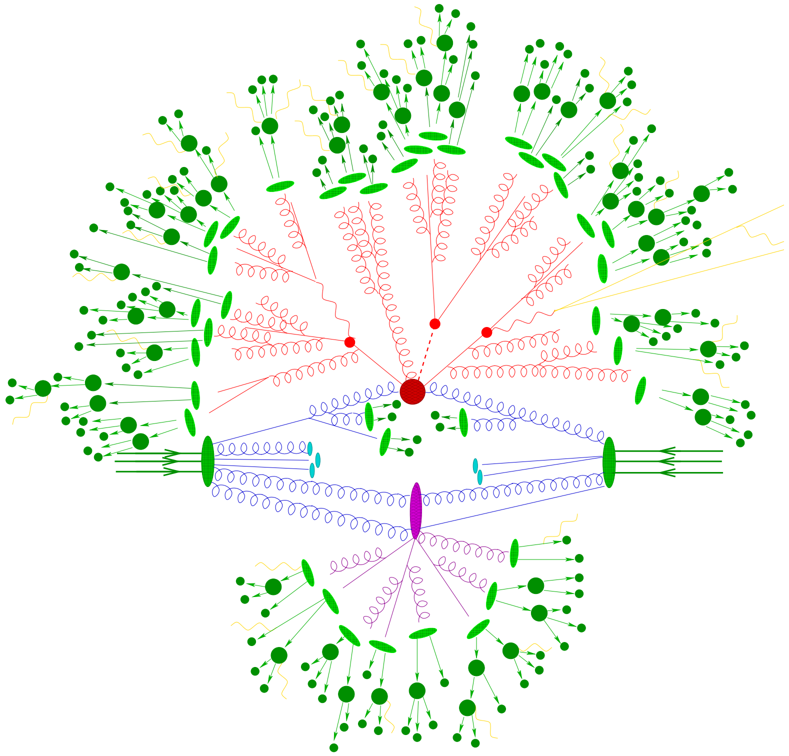
- Samples of simulated events are essential in high energy physics
- Processes at vastly different energy regimes are involved → from hard scattering to parton showering
- Luckily, this factorizes!

Hard scatter



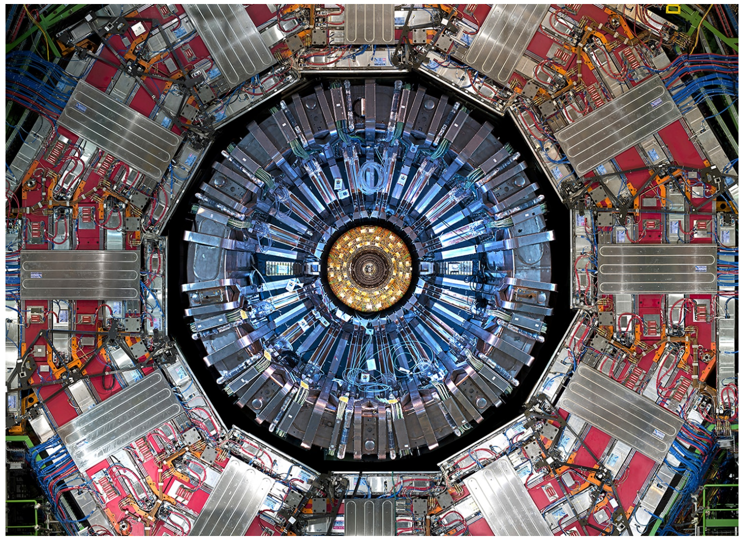
Example diagram of LO tt+Z production. *Perturbative*, MC integration of Matrix Element

Event generation



= underlying event
+ parton showering
+ hadronization
+ hadron decays
mostly non-perturbative

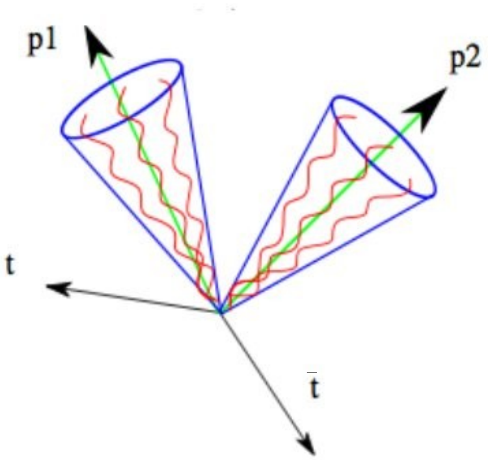
Detector simulation



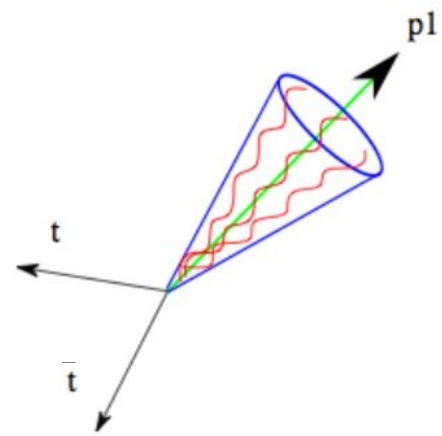
GEANT 4 (“FullSim”), Fast Simulation, Delphes

Jet Matching

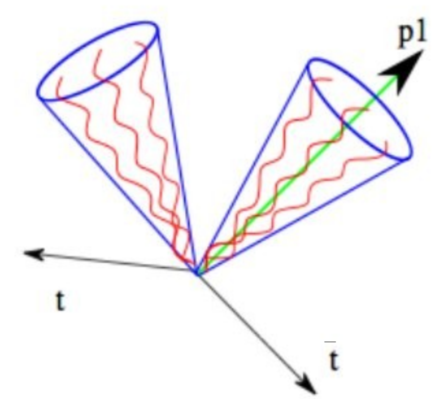
- So far, every jet in the example comes from the parton shower code (Pythia8)
- MadGraph works well in perturbative (hard / large momentum) regime, Pythia in soft regime
- Can generate the full ME for W + N jets and combine the best of both worlds
    - Problem: Double counting!



2-parton sample
(no hard PS radiation)



1-parton sample
(no hard PS radiation)



1-parton sample
+ hard PS radiation

Jet Matching

- Remember: Why do we need parton showering in the first place?
- QCD is
  - perturbative for large momentum transfers
    → Matrix element calculation works well
    (i.e. what MG5 does)
  - non-perturbative for small momentum transfers
    → Need phenomenological scale evolution approach
    (i.e. what P8 does)
- Each approach works well in one regime, underperforms/fails in the other
- Obviously question: Can we get the advantages of both? **Yes!**
- Require the k_T between two partons from MG5 to be above a threshold "xqcut", where

$$k_T = \sqrt{2 \min(p_{Ti}^2, p_{Tj}^2)[\cosh(\eta_i - \eta_j) - cos(\phi_i - \phi_j)]}$$

- Run parton shower
- After showering, jet clustering is performed and it is checked whether all jets with kT > QCUT are matched to a ME-level parton.
  - if yes, keep the event
  - if no, reject

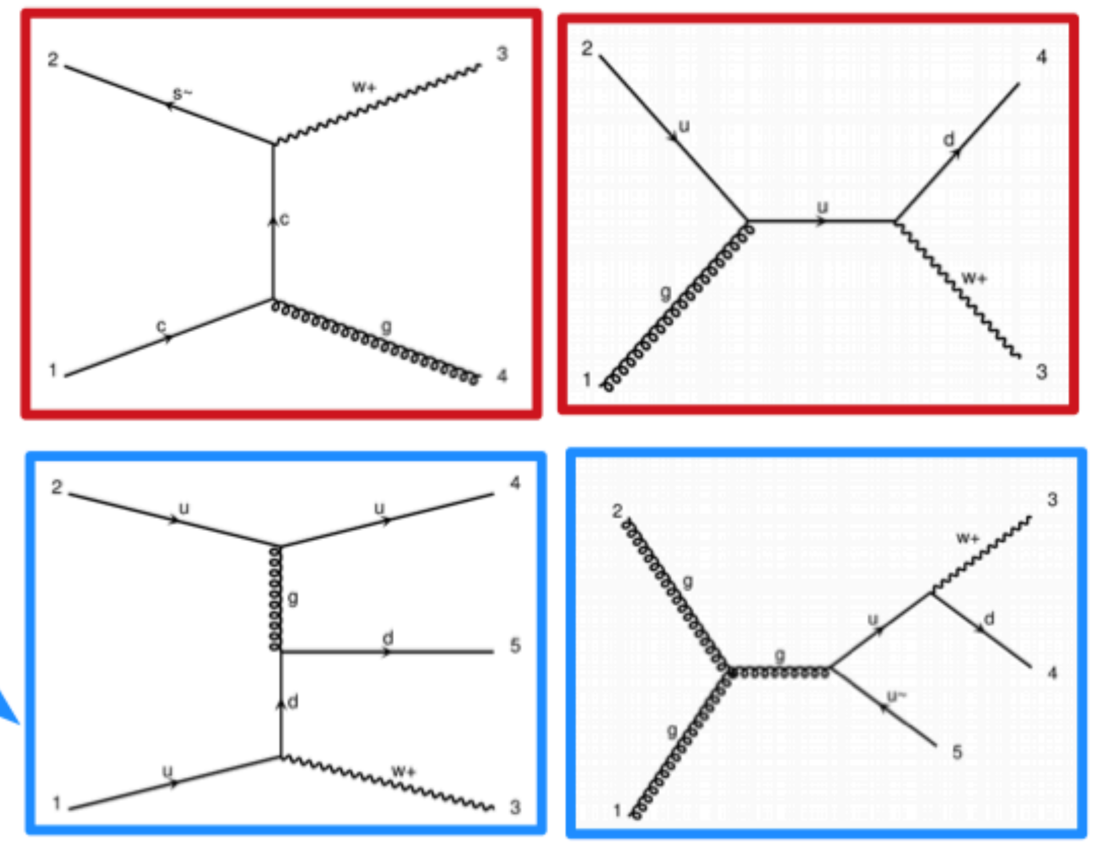
xqcut: parameter in MG run card
QCUT: parameter in Pythia; serves as "boundary" between ME and PS

W + jets example

Process card

```
Total: 3 processes with 389 diagrams
```

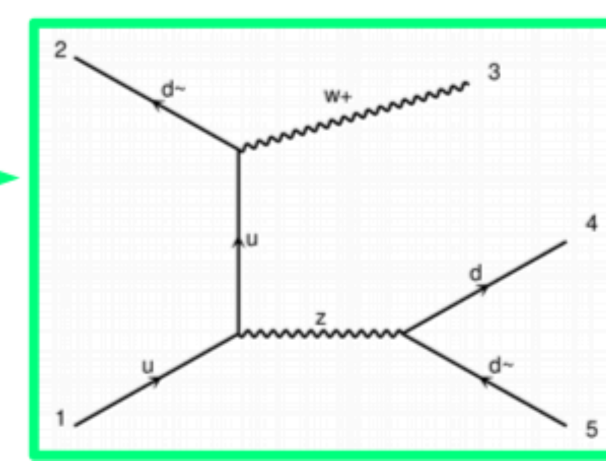
```
import model sm-ckm

define ell+ = e+ mu+ ta+
define ell- = e- mu- ta-

generate p p > w+, w+ > ell+ vl @0
add process p p > w+ j, w+ > ell+ vl @1
add process p p > w+ j j, w+ > ell+ vl @2

output wplustest_4f_012jet_LO -nojpeg
```



+ ...

+ ...

- Note that MG figured out on its own what diagrams to use.
- Notably absent: two jets from Z decay
- Unless we specifically ask for these, MG neglects them because the cross-section will be much smaller (EW instead of QCD)



W + jets example

Run card

Turn on MLM matching:

```
1         = ickkw            ! 0 no matching, 1 MLM, 2 CKKW matching
```

Cut value below which MG does not produce anything:

```
#*********************************************************************
# Jet measure cuts                                                   *
#*********************************************************************
 10   = xqcut   ! minimum kt jet measure between partons
```

Propagate xqcut threshold to ptj and mjj cuts → mostly for efficiency
(can be a matter of life and death for complicated processes)

```
#*********************************************************************
# Automatic ptj and mjj cuts if xqcut > 0
# (turn off for VBF and single top processes)
#*********************************************************************
   T  = auto_ptj_mjj  ! Automatic setting of ptj and mjj
```

Results

```
=== Results Summary for run: pilotrun tag: tag_1 ===

   Cross-section :   5.422e+04 +- 168.8 pb
   Nb of events :  0
```

LHE-level XS about a factor 2 larger than without jets!
Don't be fooled, this is mostly double counting
(i.e. you don't just get to add jets to your signal to "increase cross-section")

Matching fixes this:

```
------------------------------------
GenXsecAnalyzer:
------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------
Overall cross-section summary
-----------------------------------------------------------------------------------------------------------------------------------------------
Process         xsec_before [pb]                passed  nposw   nnegw   tried   nposw   nnegw   xsec_match [pb]                 accepted [%]     event_eff [%]
0               2.727e+04 +/- 1.840e+02         289     289     0       515     515     0       1.530e+04 +/- 6.051e+02         56.1 +/- 2.2     56.1 +/- 2.2
1               1.611e+04 +/- 1.087e+02         100     100     0       304     304     0       5.298e+03 +/- 4.355e+02         32.9 +/- 2.7     32.9 +/- 2.7
2               1.109e+04 +/- 7.484e+01         85      85      0       181     181     0       5.208e+03 +/- 4.129e+02         47.0 +/- 3.7     47.0 +/- 3.7
-----------------------------------------------------------------------------------------------------------------------------------------------
Total           5.446e+04 +/- 2.264e+02         474     474     0       1000    1000    0       2.582e+04 +/- 8.666e+02         47.4 +/- 1.6     47.4 +/- 1.6
-----------------------------------------------------------------------------------------------------------------------------------------------
Before matching: total cross section = 5.446e+04 +- 2.264e+02 pb
After matching: total cross section = 2.582e+04 +- 8.666e+02 pb
Matching efficiency = 0.5 +/- 0.0   [TO BE USED IN MCM]
Filter efficiency (taking into account weights)= (474) / (474) = 1.000e+00 +- 0.000e+00
Filter efficiency (event-level)= (474) / (474) = 1.000e+00 +- 0.000e+00    [TO BE USED IN MCM]

After filter: final cross section = 2.582e+04 +- 8.666e+02 pb
After filter: final fraction of events with negative weights = 0.000e+00 +- 0.000e+00
After filter: final equivalent lumi for 1M events (1/fb) = 3.874e-02 +- 5.037e-05
```

Jet matching performance

- Recall: we have artificially split the physical process in **energy regimes below and above a scale QCUT**

- Transition between regimes needs to be **smooth**

- Can be investigated via **Differential Jet Rates (DJR)** distributions
  - DJR corresponds to the kT separation of the final clustering step for a given jet multiplicity

    e.g. cluster event until it has 2 jets left (i.e. jet multiplicity = 2), and suppose these 2 jets have a kT separation of 20 GeV, then DJR(1→2) = 20 GeV

    Descreasing the cutoff scale from >20 GeV to <20 GeV turns event from 1-jet into 2-jet event

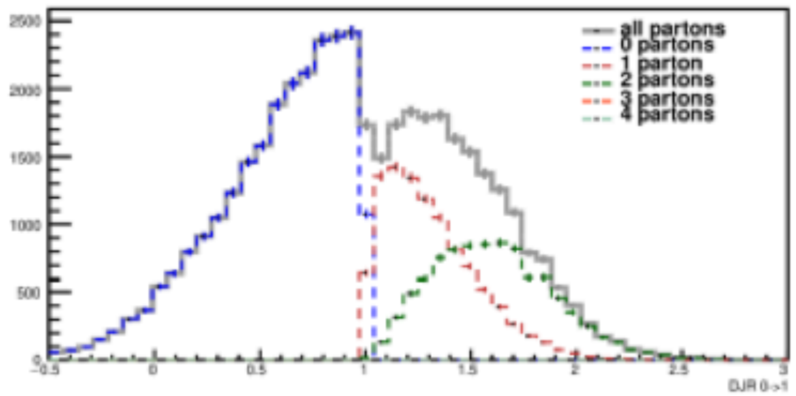
→ Goal is to **find QCUT value** that results in reasonably **smooth DJR** distributions for the sum of the contributions with different number of ME partons
illustrated in next slides

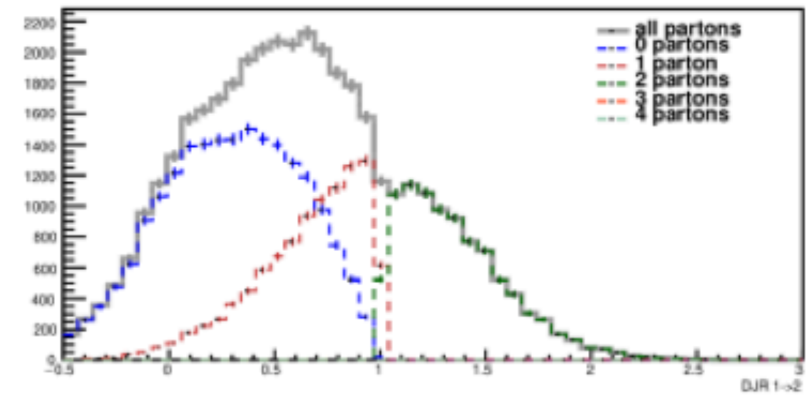
QCUT = 10 GeV

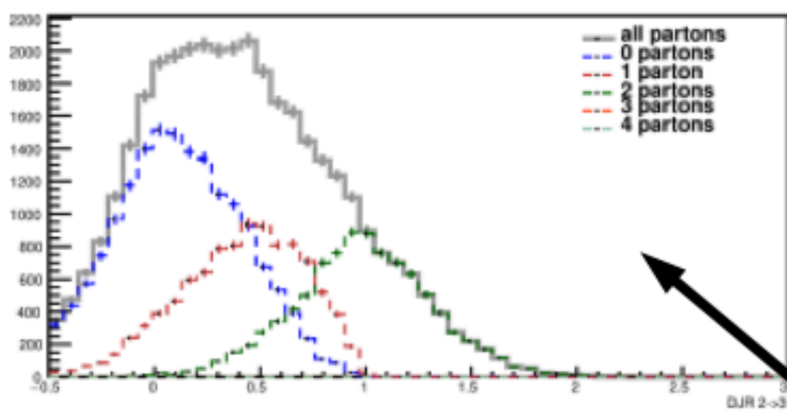
At QCUT, the lower multiplicity sample is cut off → Good

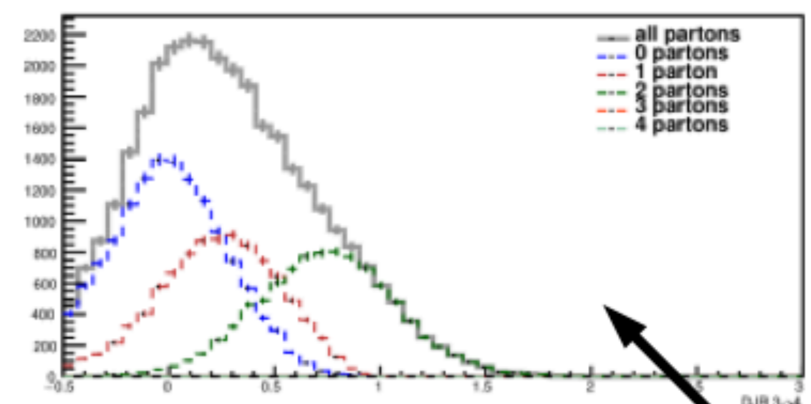
Clear discontinuity at QCUT → Bad

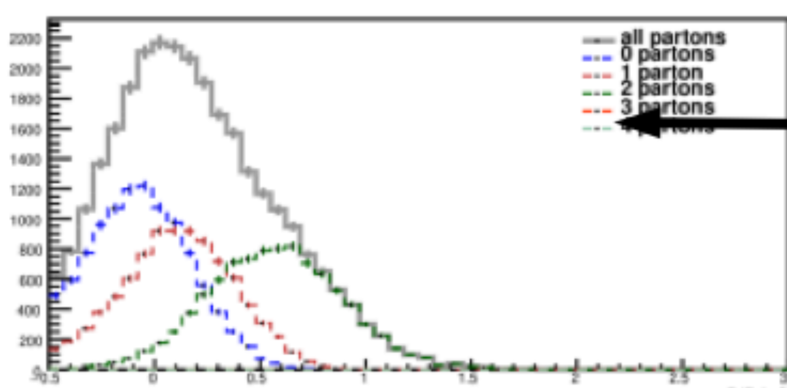
→ Try other values











Since we have a maximum of two partons in the ME calculation, the higher multiplicity plots are irrelevant here

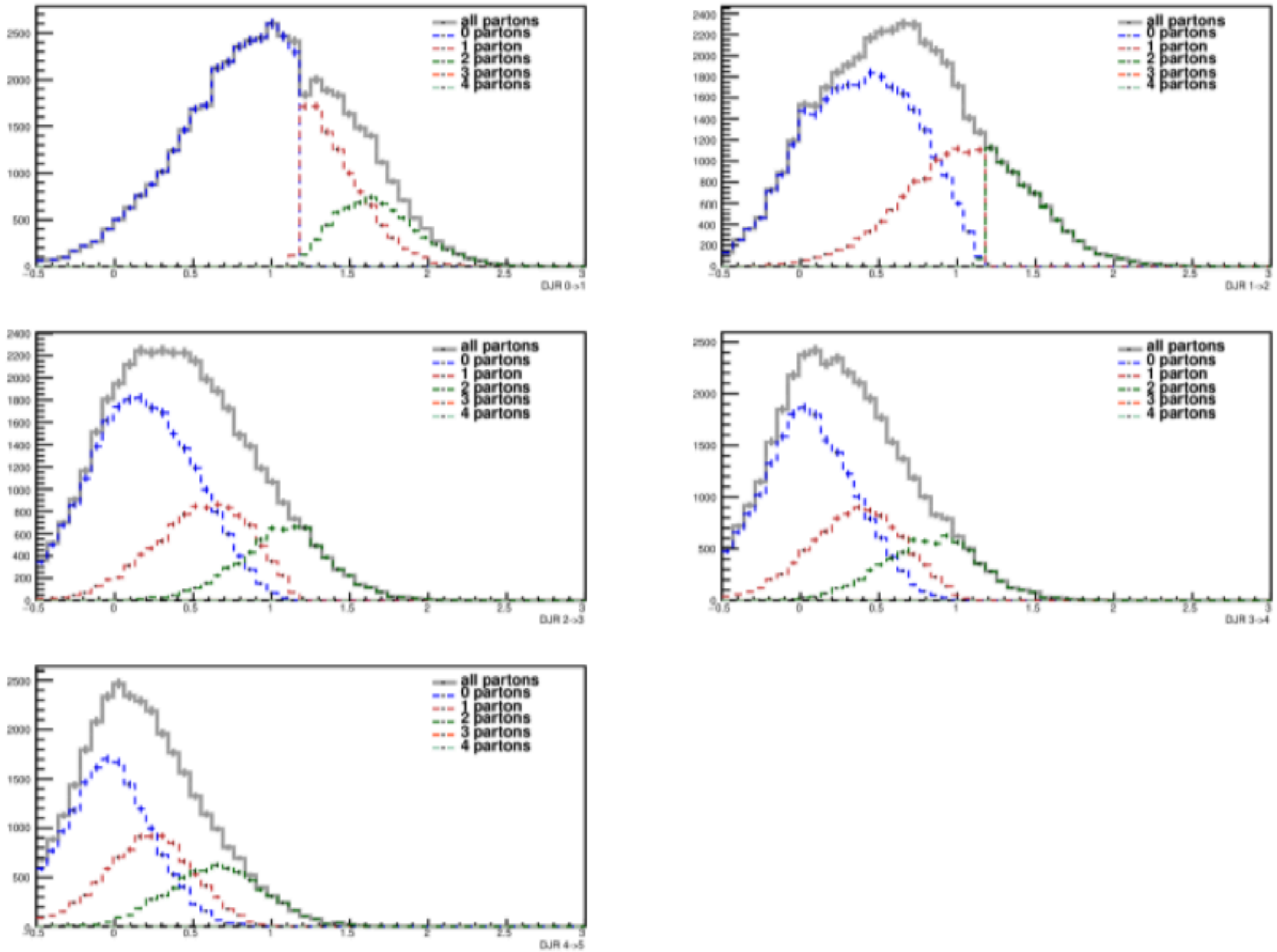
Note: x-axis is in $\log_{10}$(GeV) → QCUT=10 GeV means x=1

QCUT = 15 GeV

Note how the cut-off moved in the plot

Better, but not great

→ Keep trying



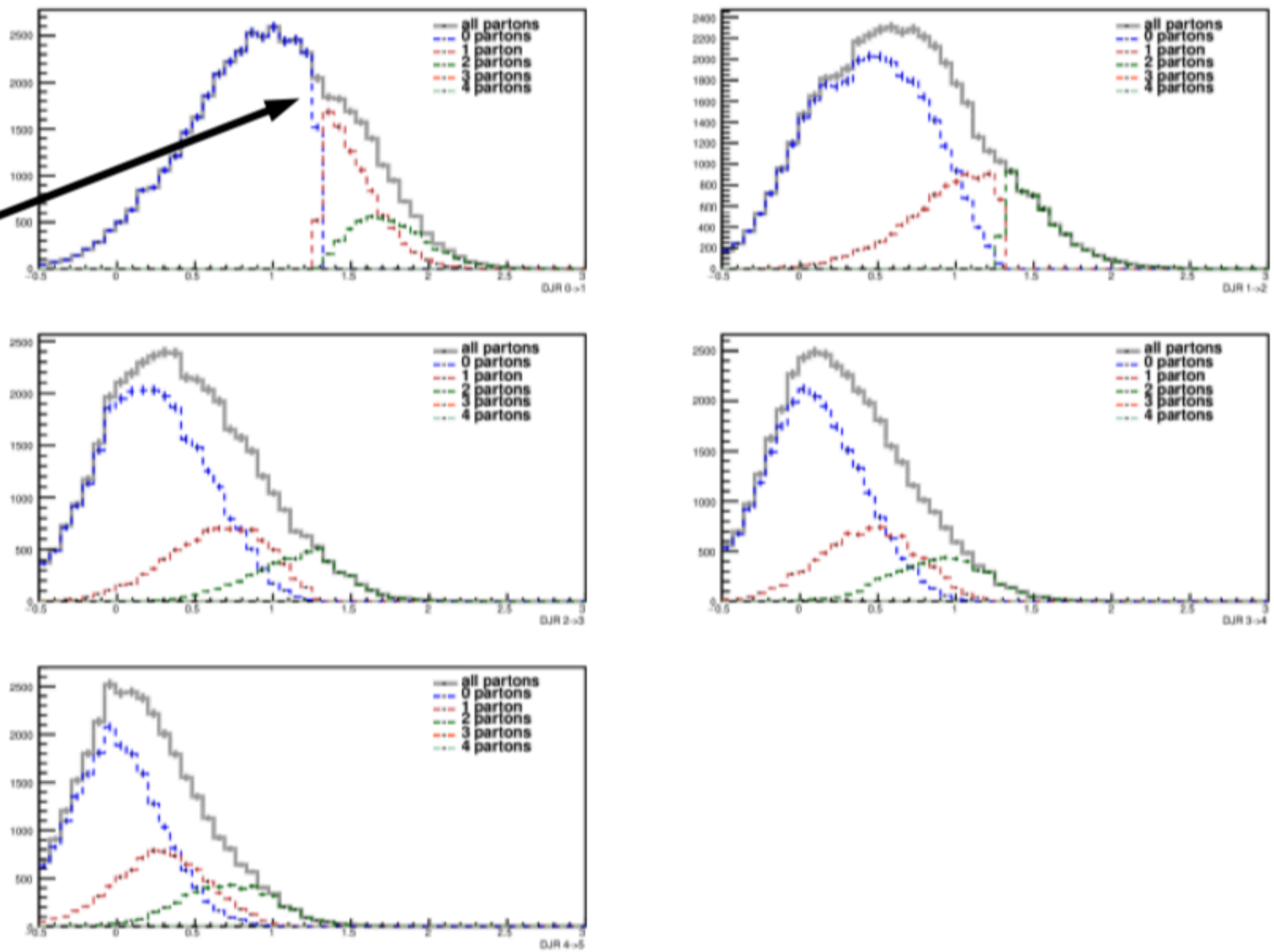
QCUT = 20 GeV

Looks better!

Still very slight kink?
May also be statistics

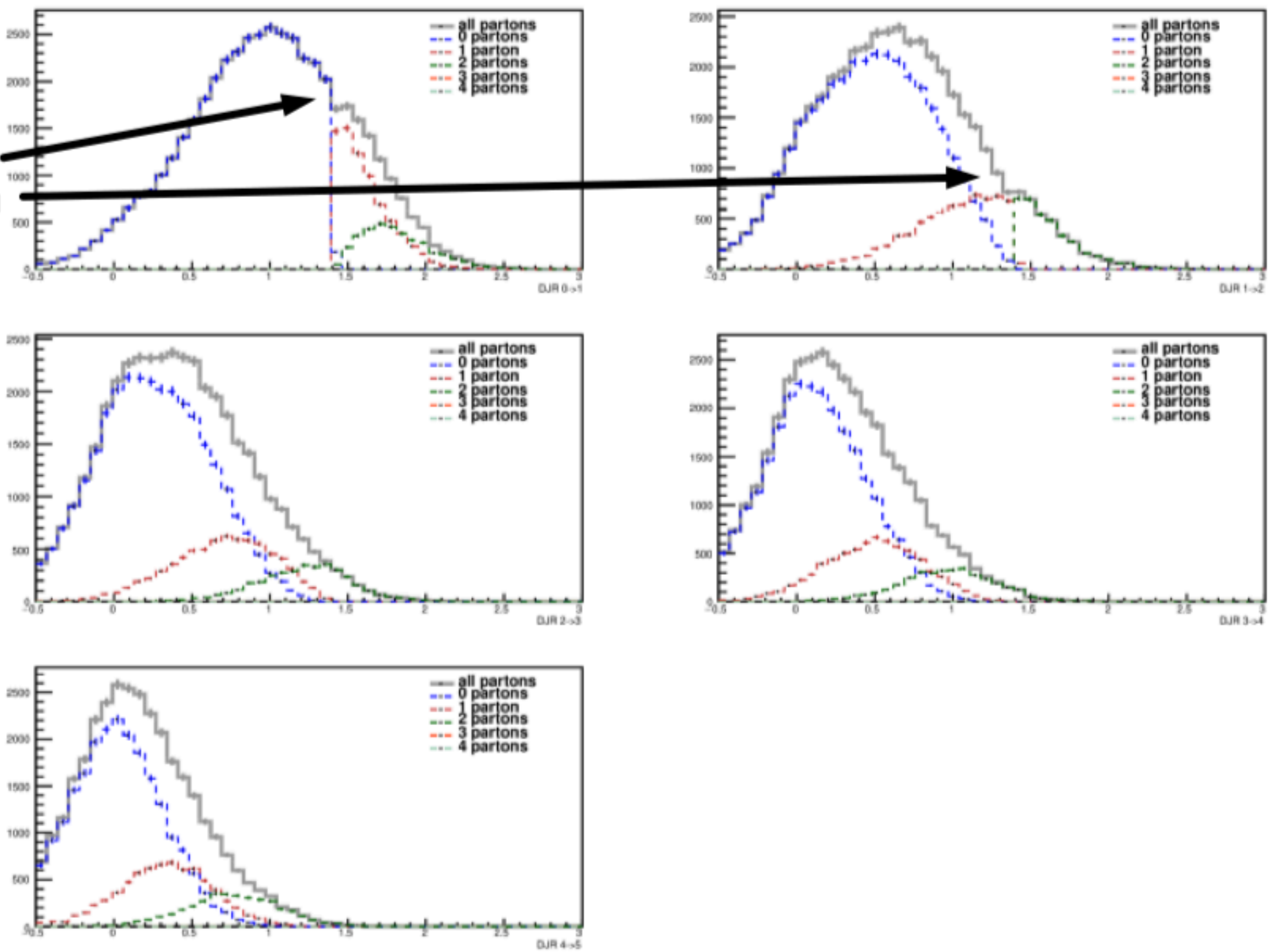
Unfortunately no hard criterion, but this level of accuracy is typically fine

If we wanted to know more accurately:
More events
+ finer QCUT scan



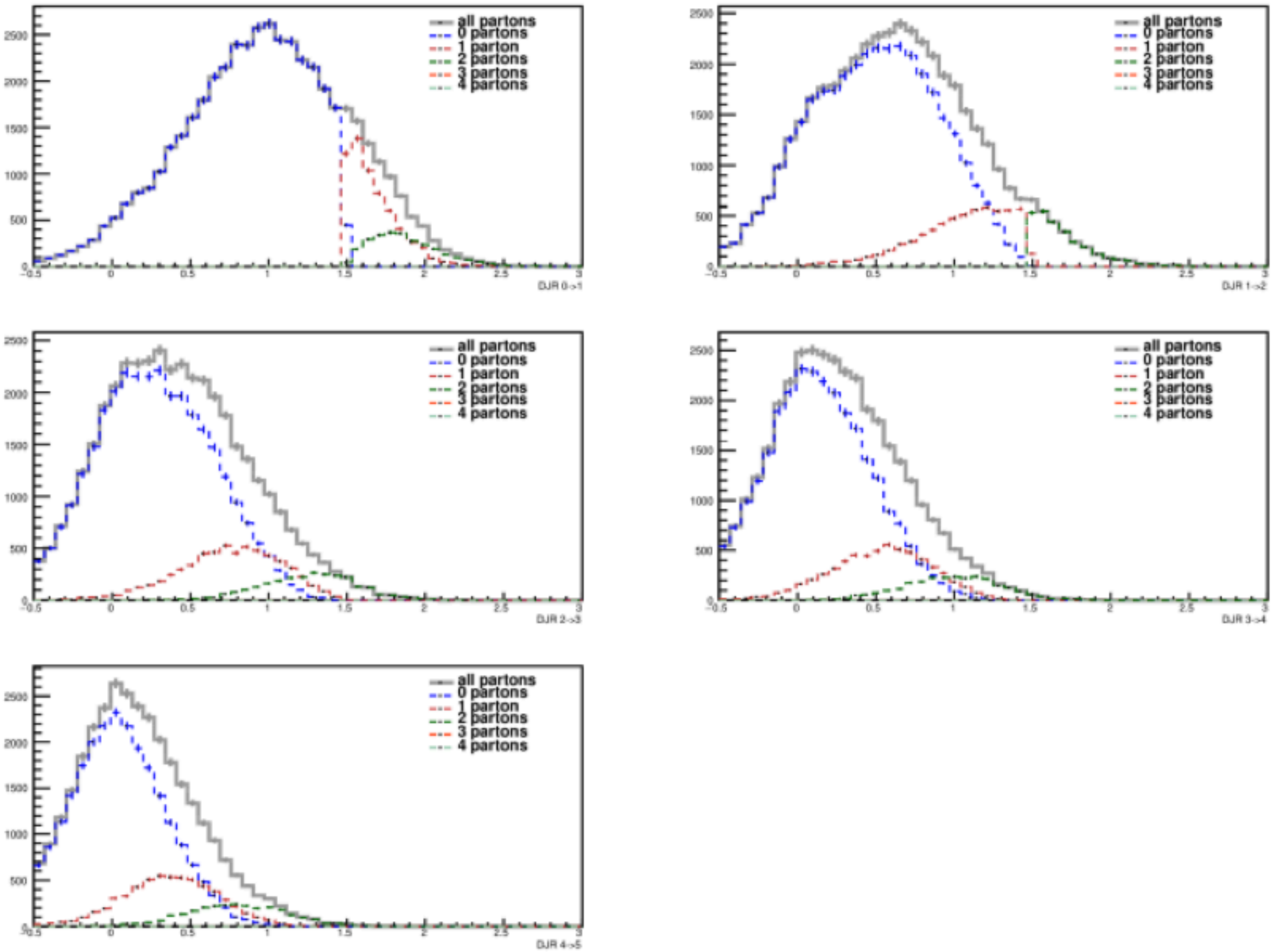
QCUT = 25 GeV

Getting worse again



QCUT = 30 GeV

Same story



QCUT = 50 GeV

Quite bumpy now

Also, consider that we used
xqcut = 10 GeV in MG

→ MG generated many events with 10-50 GeV jets; we are throwing these out now

→ Large xqcut-QCUT difference is very inefficient