BETHEL UNIVERSITY

Delphes 4 Muon Collider

JULIE HOGAN

7/13/2023

Outline

- What is Delphes?
- How delphes reads input
- Anatomy of a delphes card
- Phase-2 experience
  - Validating Delphes
  - Producing samples at scale

What is Delphes?

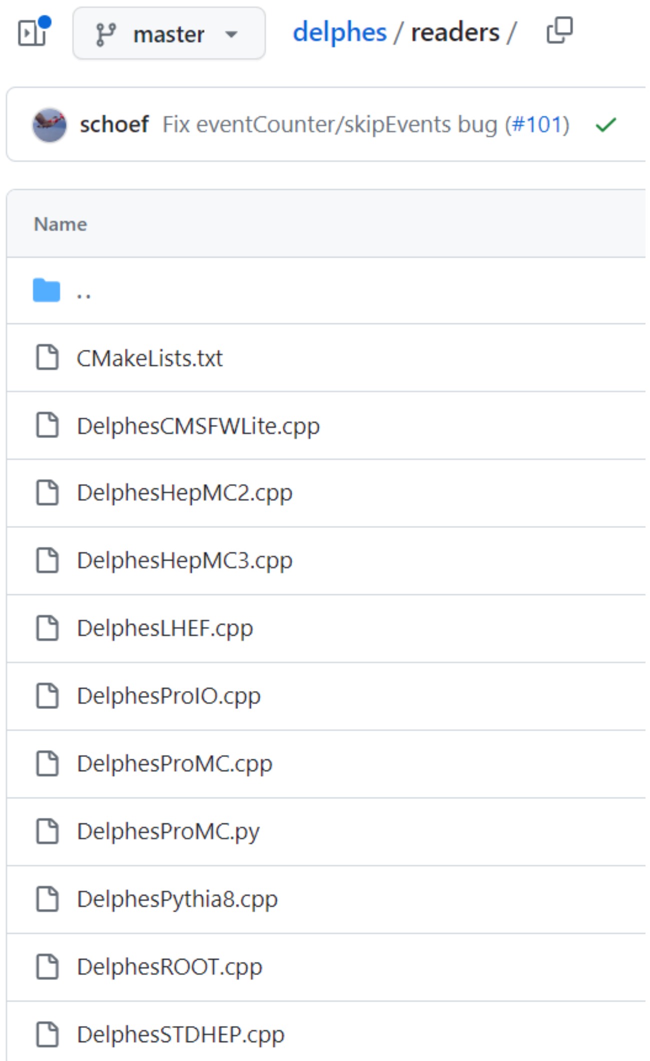
"The Delphes framwork takes as input the most common event generator output and performs a **fast and realistic simulation** of a general purpose collider detector. To do so, long-lived particles emerging from the hard scattering are propagated to the calorimeters within a uniform magnetic field parallel to the beam direction. The particle energies are computed by **smearing** the initial long-lived visible particles momenta **according to the detector resolution**. As a result, jets, missing energy, isolated electrons, muons and photons, and taus can be reconstructed."

DELPHES
fast simulation

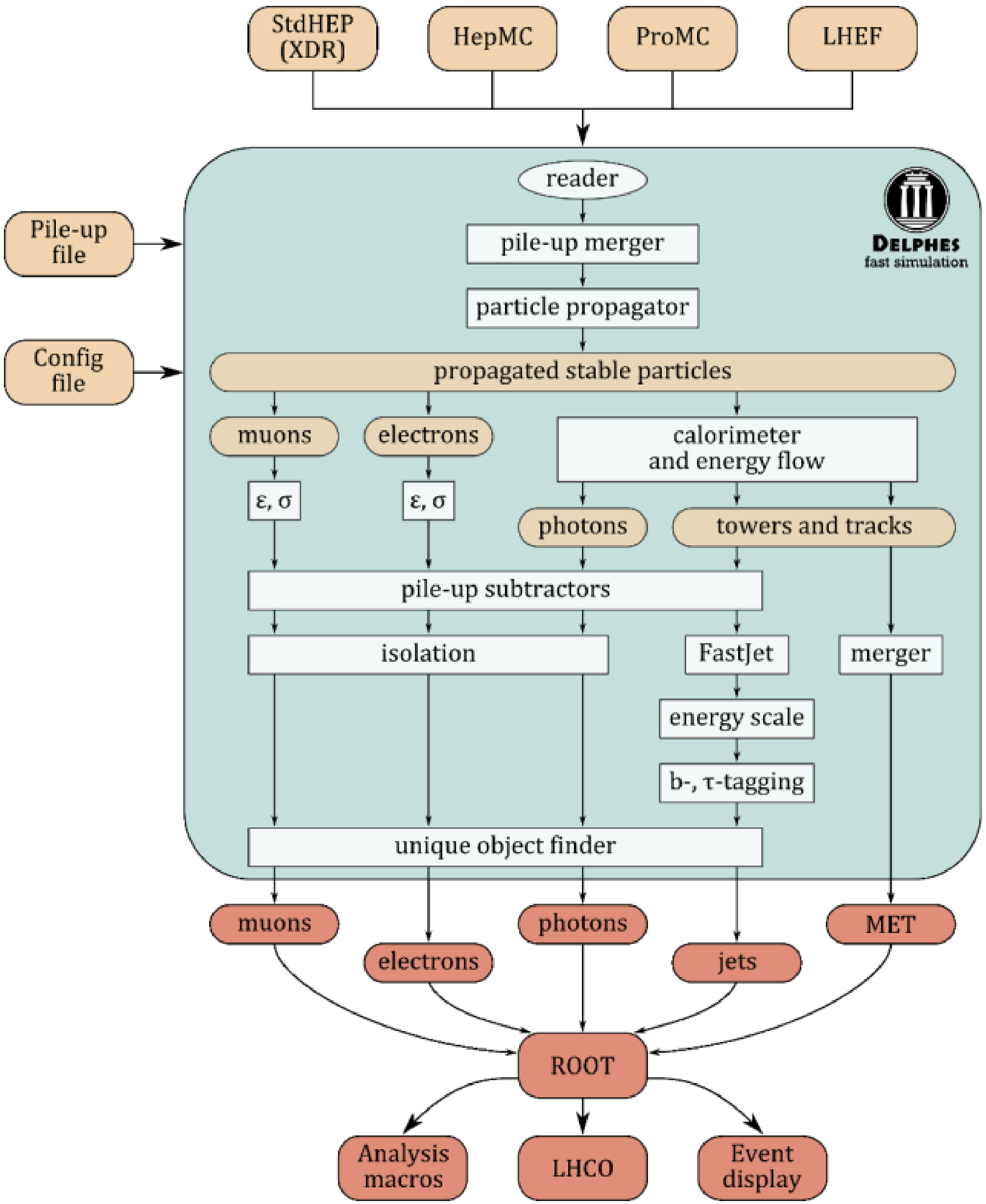
https://arxiv.org/abs/1307.6346

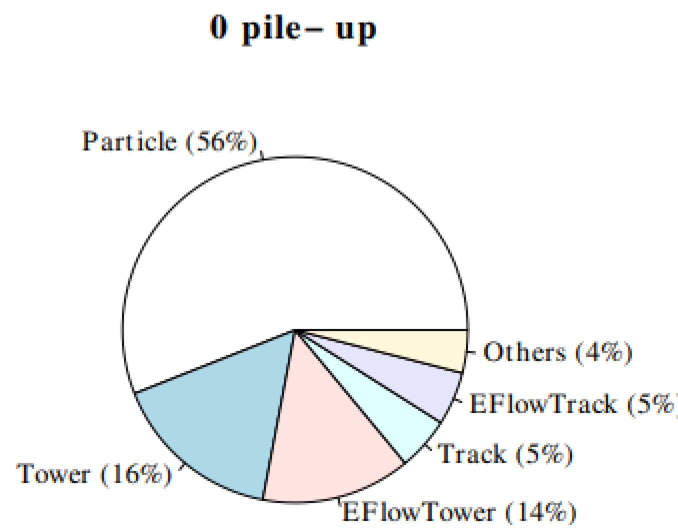
Readers

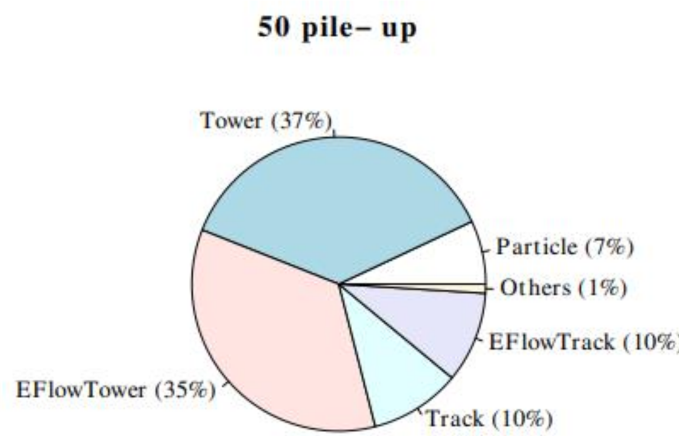
- Delphes github: https://github.com/delphes/delphes/
- Main functions are the "readers"
  - Open input files
  - Create output files
  - Create the Delphes trees
  - Pull in maxEvents / skipEvents
  - Run the event loop
  - Write the output files
- Needs LHE info and "genParticles"
- DelphesPythia8 hooks Pythia gen to delphes



Delphes analysis sequence







Modules & Classes

- Delphes cards show the list of "modules" and their order of running
- Modules take & make TObjArrays of "Candidates"
- Tcl language used to connect modules and set parameters:

`set ParamName value`
this command sets the value of the parameter given by `ParamName` to `value`

`add ParamName value value value ...`
this command treats the parameter given by `ParamName` as a list and appends each of the `value` arguments to that list as a separate element.

`module ModuleClass ModuleName ModuleConfigurationBody`
this command activates a module of class `ModuleClass` called `ModuleName` and evaluates module's configuration commands contained in `ModuleConfigurationBody`:

```
module Efficiency ElectronEfficiency {
  set InputArray ElectronEnergySmearing/electrons
  set OutputArray electrons

  # set EfficiencyFormula {efficiency formula as a function of eta and pt}

  # efficiency formula for electrons
  set EfficiencyFormula {                                      (pt <= 10.0) * (0.00) +
                                           (abs(eta) <= 1.5) * (pt > 10.0)  * (0.95) +
                         (abs(eta) > 1.5 && abs(eta) <= 2.5) * (pt > 10.0)  * (0.85) +
                         (abs(eta) > 2.5)                                   * (0.00)}
}
```

- Output of this module could be read by others as "ElectronEfficiency/electrons"

Anatomy of the card file

- ExecutionPath: the order in which modules will be executed

```
set ExecutionPath {
    ParticlePropagator
    TrackMergerProp

    DenseProp
    DenseMergeTracks
    DenseTrackFilter

    ChargedHadronTrackingEfficiency
    ElectronTrackingEfficiency
    MuonTrackingEfficiency
    ForwardMuonEfficiency

    ChargedHadronMomentumSmearing
    ElectronMomentumSmearing
    MuonMomentumSmearing
    ForwardMuonMomentumSmearing

    TrackMerger

    ECal
    HCal

    Calorimeter
    EFlowMerger
    EFlowFilter
```

…skipping 300 modules for jets of EVERY radius and btags of EVERY working point for EVERY radius…

```
        TauTagging_R12N5
        TauTagging_R12N6
        TauTagging_R15N2
        TauTagging_R15N3
        TauTagging_R15N4
        TauTagging_R15N5
        TauTagging_R15N6
        TauTagging_R02_inclusive
        TauTagging_R05_inclusive
        TauTagging_R07_inclusive
        TauTagging_R10_inclusive
        TauTagging_R12_inclusive
        TauTagging_R15_inclusive

        ScalarHT

        TreeWriter
    }
```

Anatomy of the card file

- Link modules together based on their inputs and outputs:

```
##################################
# Propagate particles in cylinder
##################################

module ParticlePropagator ParticlePropagator {
  set InputArray Delphes/stableParticles

  set OutputArray stableParticles
  set ChargedHadronOutputArray chargedHadrons
  set ElectronOutputArray electrons
  set MuonOutputArray muons

  # radius of the magnetic field coverage in t
  set Radius 1.5
  # half-length of the magnetic field coverage
  set HalfLength 2.31

  # magnetic field, in T
  set Bz 4.0
}
```

This array is created by the "Reader" from the generator input

```
fOutputArray->Add(candidate);

if(TMath::Abs(q) > 1.0E-9)
{
  switch(TMath::Abs(candidate->PID))
  {
  case 11:
    fElectronOutputArray->Add(candidate);
    break;
  case 13:
    fMuonOutputArray->Add(candidate);
    break;
  default:
    fChargedHadronOutputArray->Add(candidate);
  }
}
```

```
##############
# Track merger
##############

module Merger TrackMergerProp {
# add InputArray InputArray
  add InputArray ParticlePropagator/chargedHadrons
  add InputArray ParticlePropagator/electrons
  add InputArray ParticlePropagator/muons
  set OutputArray tracks
}
```

Anatomy of the card file

- Different types of modules:

- Mergers – seen already
- Efficiencies / Resolutions: apply random filtering or random smearing based on provided formulas
- Calorimeter: declare the granularity of a calorimeter. Designed to work in a "particle flow" model returning "tracks" and "towers"
- Isolation: given some "eFlow" category, compute isolation from other eFlow candidates
- JetFinder: clusters eFlow candidates into jets!

- Can link other tcl files! Example: VLC jet finder

Running delphes

- Installation options in Github README
- Needs some tweaks if you want to borrow CMS simulation: DelphesCMSFWLite doesn't compile unless CMSSW is found

```
cmsrel CMSSW_10_0_5
cd CMSSW_10_0_5
cmsenv
cd ..
git clone https://github.com/delphes/delphes.git
cd delphes
./configure
sed -i -e 's/c++0x/c++1y/g' Makefile
make -j 10
```

- ./DelphesReaderOfChoice cards/myCard.tcl OUTFILE INFILE

```
[jmanagan@cmslpc125 delphes]$ ./DelphesCMSFWLite cards/delphes_card_MuonColliderDet.tcl ttbarMuCol.root
root://cmsxrootd.fnal.gov//store/mc/RunIIAutumn18MiniAOD/TTToSemiLeptonic_TuneCP5_13TeV-powheg-
pythia8/MINIAODSIM/102X_upgrade2018_realistic_v15-v1/00000/0656B91C-5F47-8A45-A9C1-AA5C2BF2F506.root
```

Some CMS experience

BETHEL UNIVERSITY

OLD SLIDES AHEAD!

For TDRs + 2018 Yellow Report we generated 4B delphes events!

Then again for 2021 Snowmass…this time with lots of validation w.r.t Phase 2 fullsim

One way to run lots of delphes…

BETHEL UNIVERSITY

- Overview of the job:

```
# Copy and unpack the tarball
echo "xrdcp source tarball and pileup file"
xrdcp -f root://eoscms.cern.ch//store/group/upgrade/RTB/delphes_tarballs/Delphes350_NtuplizerV0.tar tarball.tar
```

```
echo "Running delphes with DelphesNtuplizer/cards/$CARD"
./DelphesCMSFWLite ../cards/$CARD ${FILEOUT} ${FILEIN}
```

```
echo "Running Delphes Ntuplizer on $FILEOUT to produce $NTUPLE"
python ../bin/Ntuplizer.py -i $FILEOUT -o $NTUPLE
```

```
# copy output to eos
echo "xrdcp -f ${FILEOUT} root://${URL}/${OUTPUT}/${FILEOUT}"
```

- Gen2Delphes Repository
  - For FNAL or CERN condor
  - For FNAL or CERN storage
- Your work:
  - Set up your list of samples
  - Submit jobs
  - Run the error checker
  - Resubmit jobs

RTB Gen2Delphes -- Snowmass 2021

These scripts facilitate submitting HTCondor jobs that process a defined set of GEN input files through Delphes.

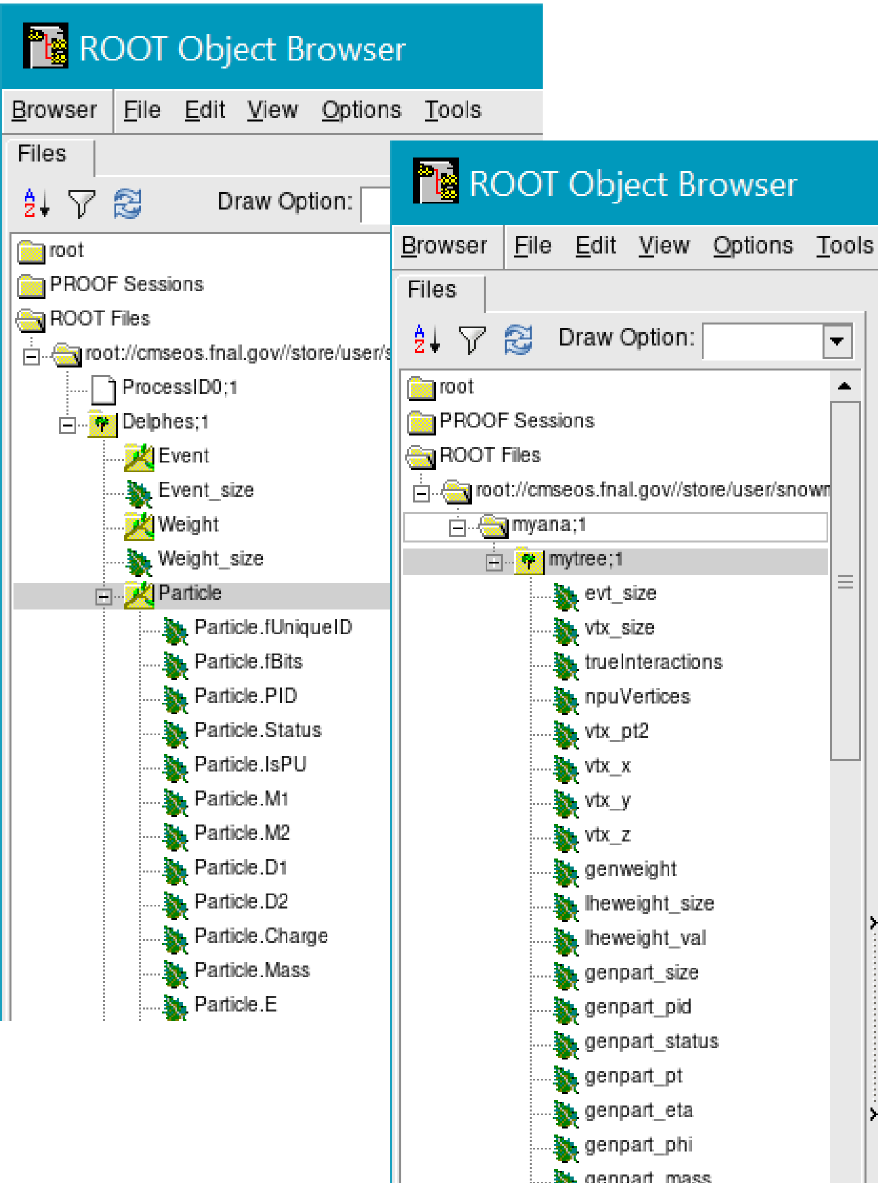
- Current CMSSW = CMSSW_10_0_5
- Current Delphes tag = 3.5.0
- Current Delphes card **from DelphesNtuplizer** = CMS_Phase2_200PU_Snowmass2021_v0.tcl

Production Spreadsheet! Google Sheets

Overview of the important scripts

- `submitCondor.py` is the main submitter that you will run. Arguments:
  - condor site (REQUIRED): FNAL or CERN, choose where you will launch condor jobs
  - storage site (REQUIRED): FNAL or CERN, choose where you will store files

Delphes Ntuples



- The NanoAOD to Delphes's AOD!
- Made with DelphesNtuplizer
- Simple way to parse the Delphes classes
  - General Info: https://twiki.cern.ch/twiki/bin/view/CMS/DelphesInstructions
  - Event loop analysis: ntuple_example.py in ValidationTools repository
  - Bamboo Analysis (RDataFrame): https://gitlab.cern.ch/cp3-cms/bamboo

Validating CMS Delphes

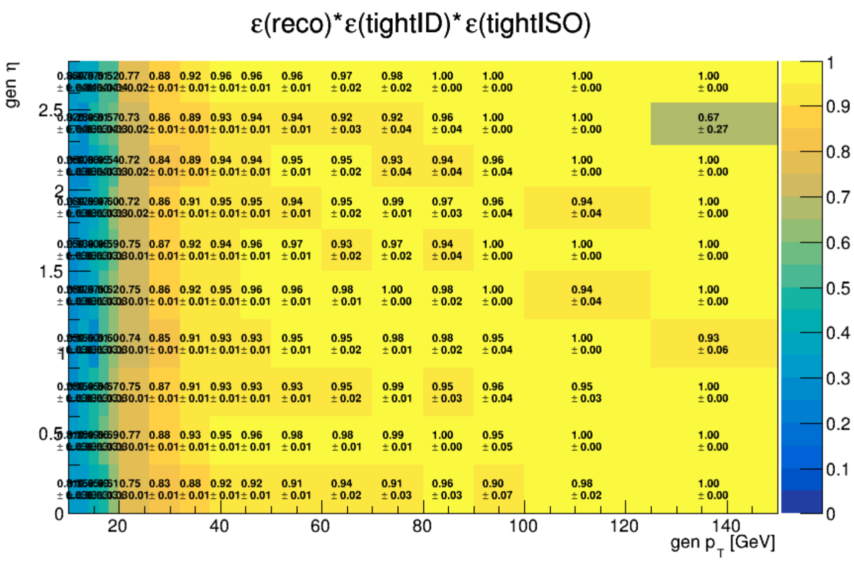
BETHEL UNIVERSITY

- Delphes has been extensively validated for Phase 2
  - Electrons, photons, muons, jets: eff, fake rate, scale/resolution for loose/med/tight
  - Tau tagging, b tagging with several working points
- Final card is part of the DelphesNtuplizer repository
- Instructions for using Delphes files and the flat trees: https://twiki.cern.ch/twiki/bin/view/CMS/DelphesInstructions

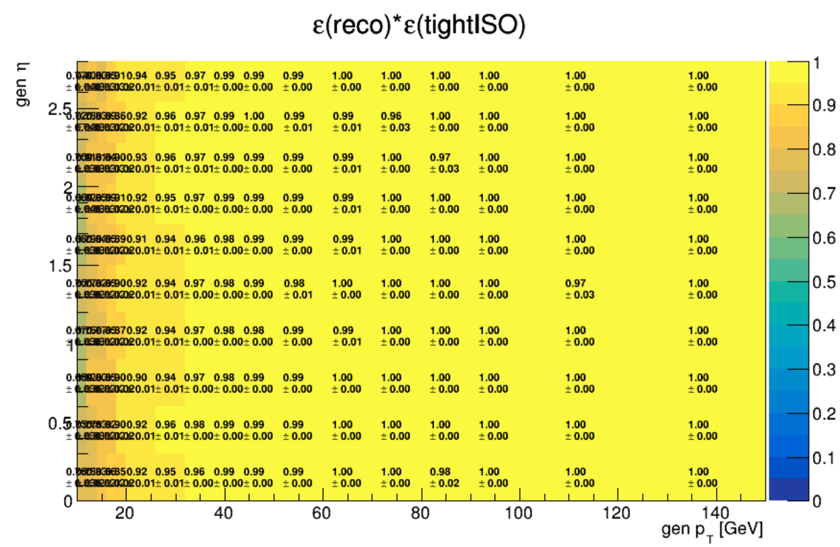
- Process:
  - Created a “dummy“ delphes card with mostly flat / default settings
  - Run samples of Delphes and fullsim through the framework
  - Create a “validated” card with observed parameterizations → Repeat for closure

Validating Delphes

- Efficiency = fraction of gen objects matched to a reco object
  - Parameterize any Delphes/fullsim efficiency disagreement into Delphes "ID"
  - reconstruction & isolated baked in to Delphes
  - Delphes eff(ID) = Fullsim eff(Reco * ID * Iso) / Delphes eff(Reco * Iso)

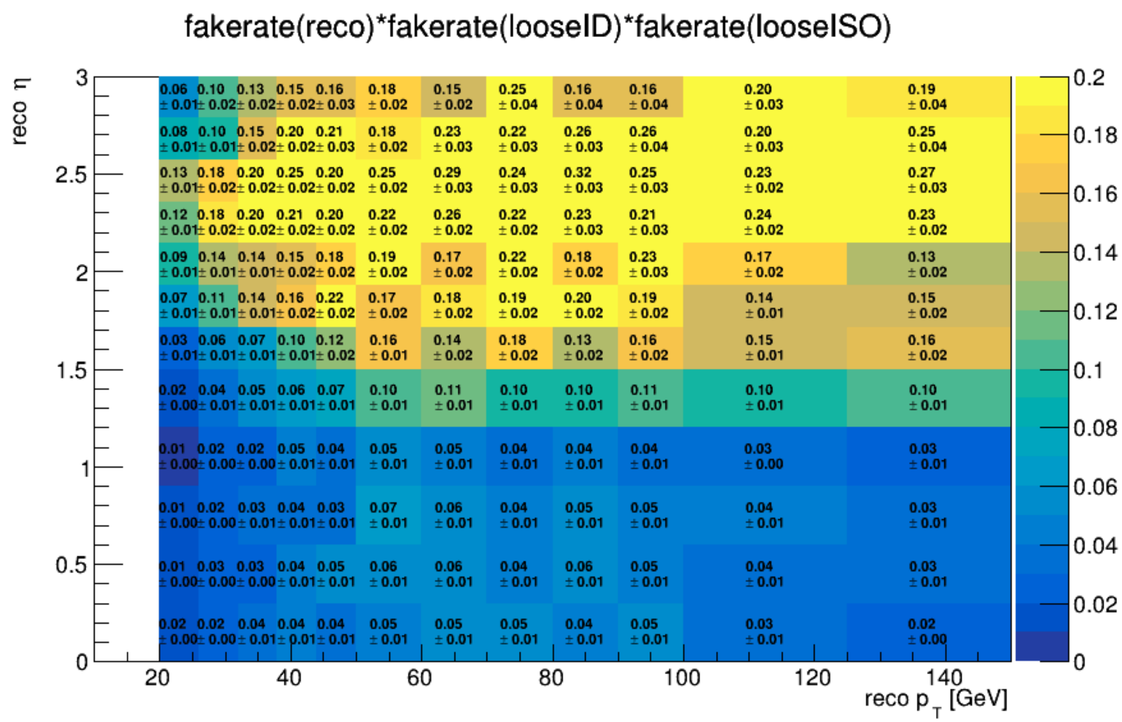


/



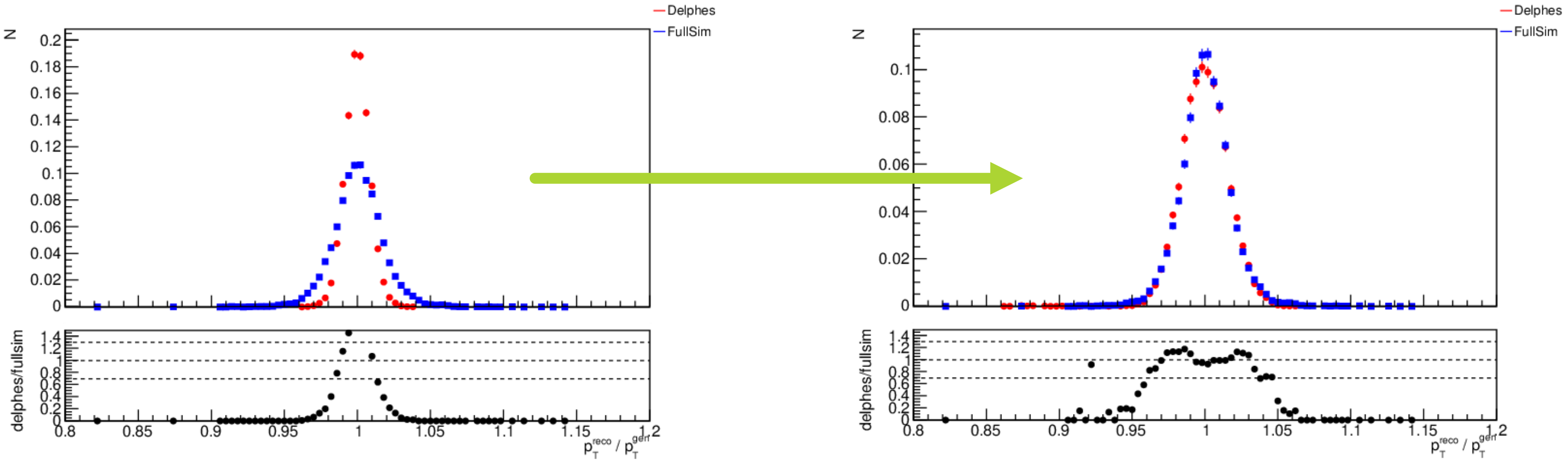
Validating Delphes

- Fakerate = fraction of “fake” objects (jets) wrongly labeled as lepton/photon
  - Use all reco jets with basic pt/eta bounds
  - Different lepton/photon qualities
  - Includes effect from pileup jets with no gen info



Validating Delphes

- Resolution = transverse momentum scale and smear
  - Delphes will be scaled by $\mu(\text{fullsim})/\mu(\text{delphes})$
  - Delphes will be smeared by $\sigma(\text{smear}) = \sqrt{\sigma(\text{fullsim})^2 - \sigma(\text{delphes})^2}$

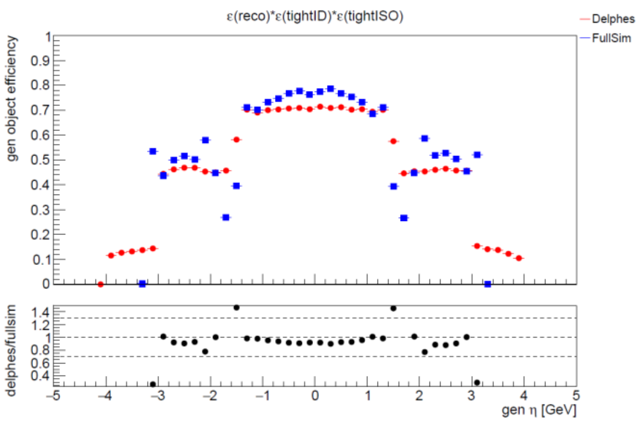


- Pt Response = distributions of reco / gen momentum
  - No direct parameterization, but tests the other tuning results

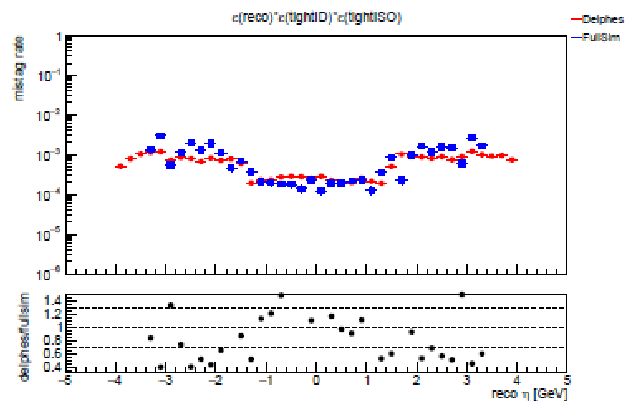
Electrons – tight

Effic.	Fakes	Scale	Smear
✓	✓	✓	

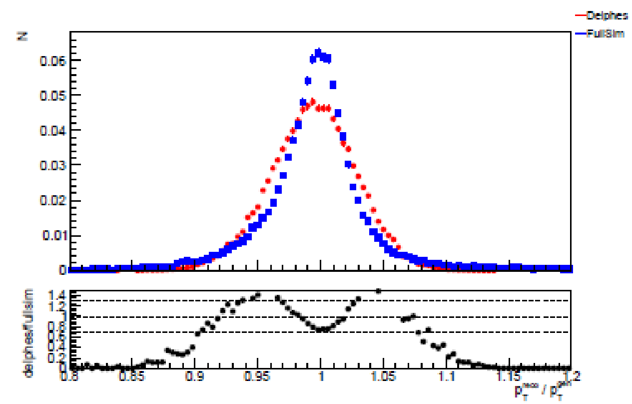
- Tight electrons: good efficiency & fakerate, some over-smearing to investigate
  - Eta tuning bins are coarse



Efficiency

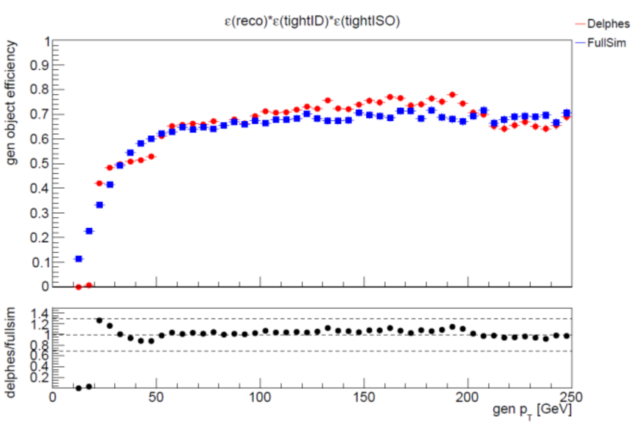


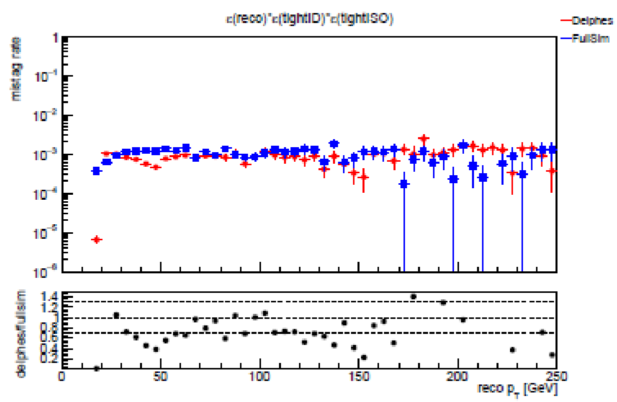
Fakerate



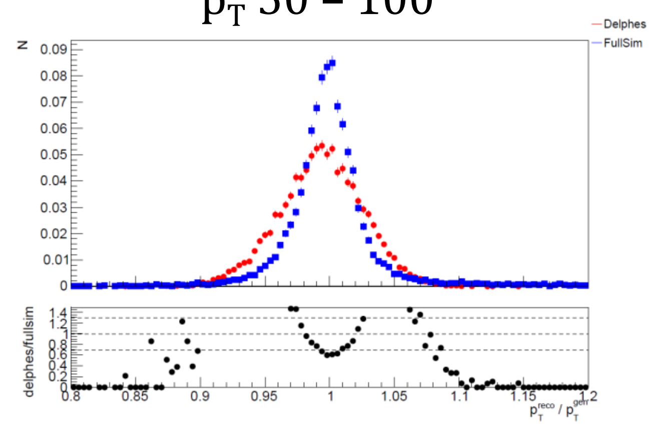
$0 < |\eta| < 1.5$ and $20 < p_T < 50$

Resolution |η| < 1.5





p_T 50 – 100



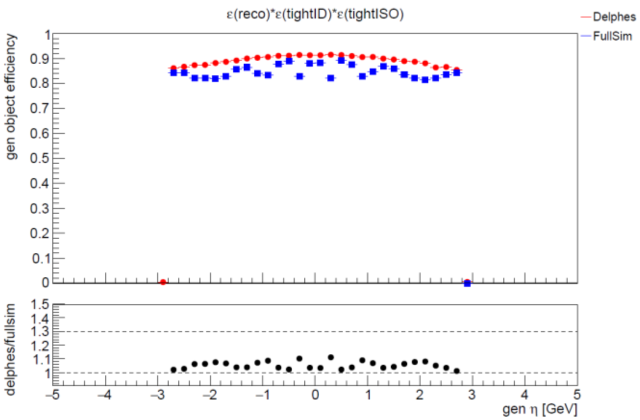
Muons – tight

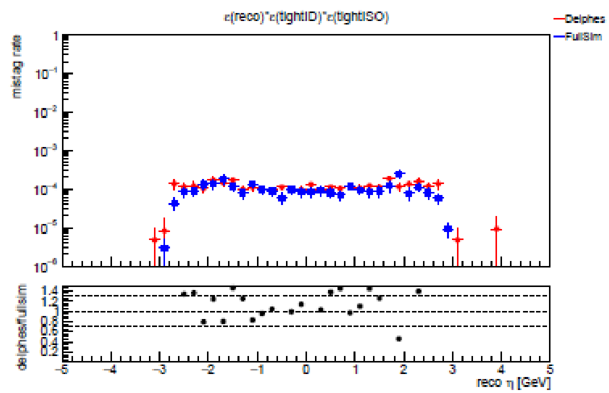
Effic.	Fakes	Scale	Smear
✓	✓	✓	✓

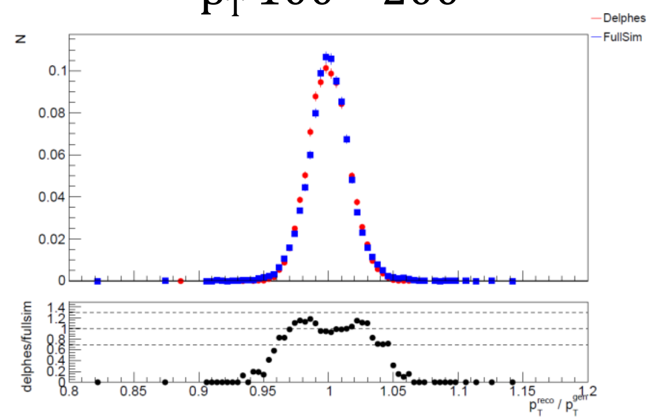
BETHEL UNIVERSITY

- Tight muons: efficiency better here. Fakerates agree given bins.
- Scale/resolution generally quite good up to 200 GeV

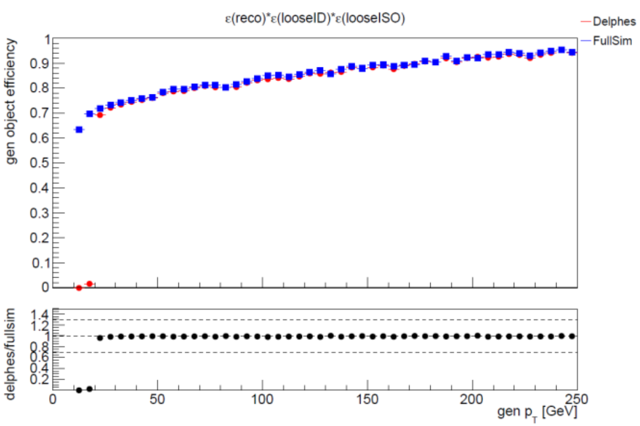
p_T 100 – 200

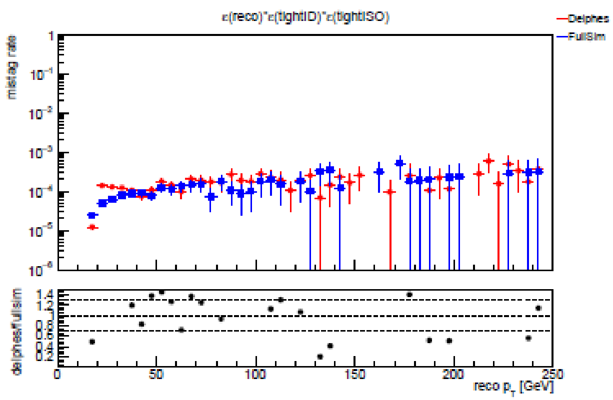


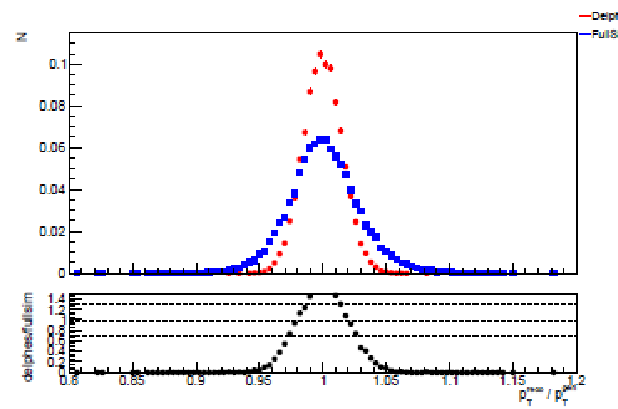




Efficiency **Fakerate** **Resolution |η| < 1.5**





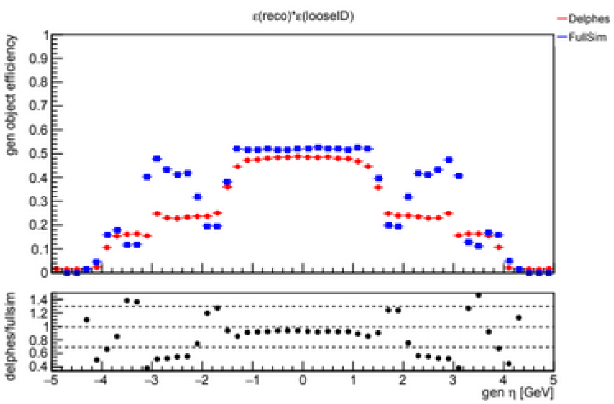


$0 < |\eta| < 1.5$ and $200 < p_T < 500$

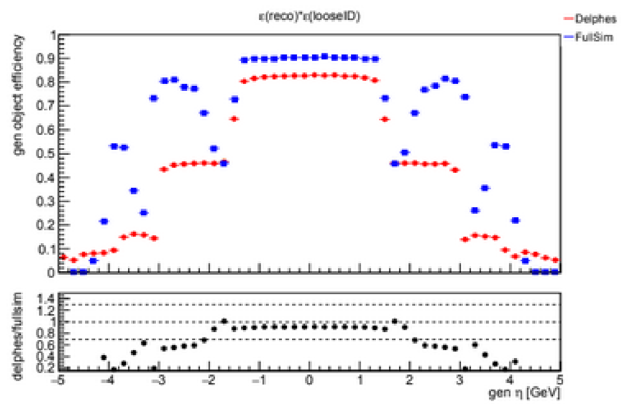
Jets

Effic.	Fakes	Scale	Smear

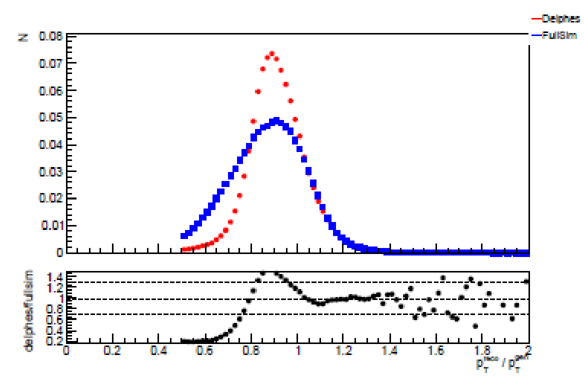
- Jets: some over-correction of efficiency, interplay with low scale
- Matching smear gets tough – plan an intermediary card for closer initial guess



$20 < p_T < 50$

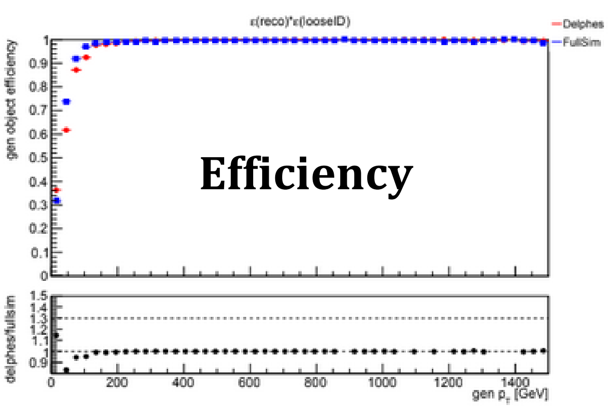


$50 < p_T < 100$

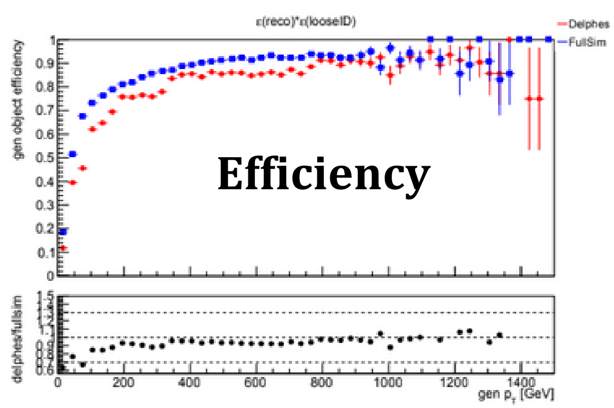


$0 < |\eta| < 1.5$ and $50 < p_T < 100$

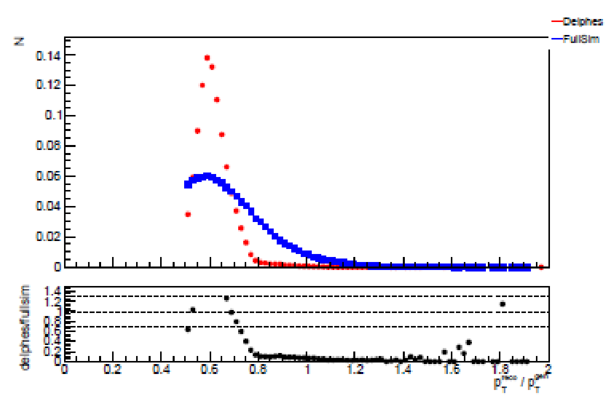
Resolution



$0 < |\eta| < 1.5$



$1.5 < |\eta| < 3.0$

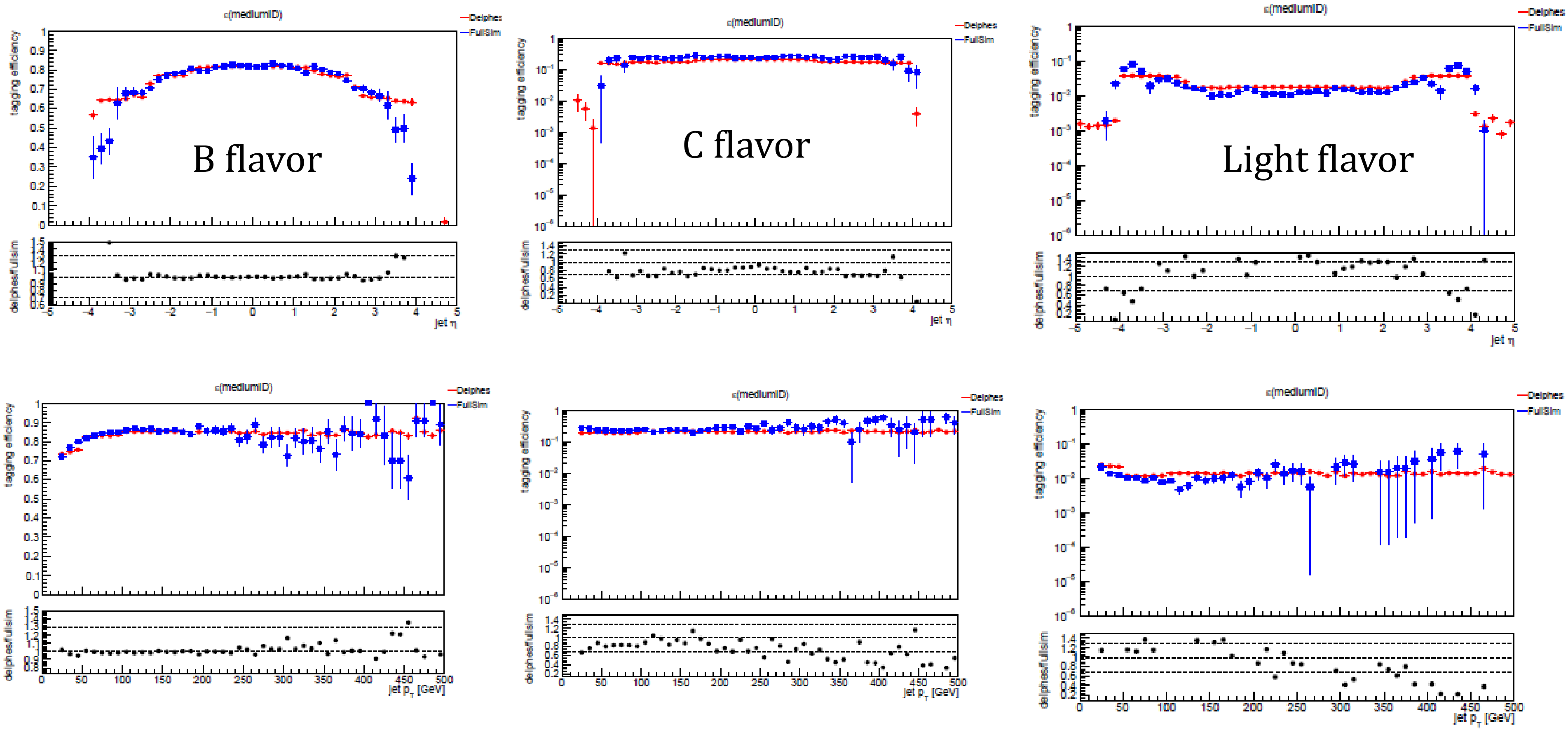


$1.5 < |\eta| < 3.0$ and $50 < p_T < 100$

Jets: b-tagging

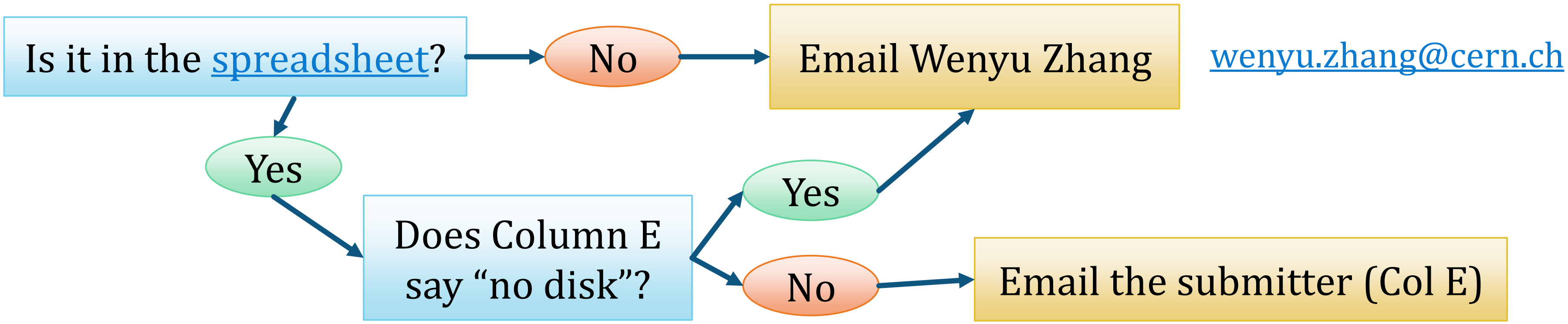
Effic.	Fakes	Scale	Smear
✓	✓		

- Medium b tags – looking nice! Also for loose and tight WPs



How can I use the samples?

- What if I can't find my sample?



(full names are on title slide)

- How do I find the ROOT files stored at FNAL?
  - `eos root://cmseos.fnal.gov/ ls -l /store/user/snowmass/Snowmass2021/`
- How do I find the ROOT files stored at CERN?
  - `eos root://eoscms.cern.ch/ ls -l /store/group/upgrade/Snowmass2021/`
- How do I find the ROOT files stored at TIFR?
  - `xrdfs se01.indiacms.res.in ls /cms/store/group/Snowmass_2021_2022/`
  - Soumya has put file lists at `/afs/cern.ch/work/m/mukherje/public/Snowmass`
  - Contact soumya.mukherjee@cern.ch if you can't find the list for a TIFR sample

How can I use the samples?

- What if the cross section is wrong or missing?
  - Our sheet was copied from the YR sheet! Most Snowmass samples lack xsecs
  - **Please help us get these right!** You can edit the spreadsheet
- How do the Delphes flat trees work?
  - https://twiki.cern.ch/twiki/bin/view/CMS/DelphesInstructions
  - Contact Michele: Michele.Selvaggi@cern.ch
- How can I analyze Delphes files or the Delphes flat trees?
  - Event loop analysis: ntuple_example.py in ValidationTools repository
  - Bamboo Analysis (RDataFrame): https://gitlab.cern.ch/cp3-cms/bamboo