How to flat-tree with NUISANCE

UNIVERSITY of ROCHESTER

Clarence Wret

Pittsburgh Tensions Workshop 2019
10 July 2019

Introduction

- Flat-trees provided for GENIE, NuWro and NEUT
  - Thanks to Steven, Andy, Jan, Hayato-san, Steve, Callum involved in generating and flat-ifying!
- Don't need generators set up: just need ROOT (v5, v6 will do)
- Flat-trees contain final state information, theory variables (e.g. q_0), interaction mode, selection flags, cross-section scaling, weights, etc
- Can easily compare generator against generator using different selections

How to get files

- Moved from Box to FNAL DUNE GPVMs in 5 minutes yesterday
- Either use mine or your own

```
[cwret@dunegpvm05 flattened]$ pwd
/pnfs/dune/persistent/users/cwret/Pittsburgh/flattened
```

- If getting your own, cd somewhere you want the files and do

```
lftp -c 'open -u tensions@pitt.edu,10sions$19 ftps://ftp.box.com; \
        set ftp:ssl-protect-data true; \
        set ftp:ssl-force true; \
        set ftp:ssl-auth TLS; \
        set ssl:verify-certificate no; \
        mirror --parallel=5 flattened'
```

- Opens up 5 simultaneous transfers, change if needed

Looking inside the trees

```
[cwret@dunegpvm05 nuwro]$ root -l nuwro_microboone_numu_Ar_2mega_NUISPREP_flat.root
root [0]
Attaching file nuwro_microboone_numu_Ar_2mega_NUISPREP_flat.root as _file0...
_froot [1] _file0.ls()
TFile**     nuwro_microboone_numu_Ar_2mega_NUISPREP_flat.root
 TFile*     nuwro_microboone_numu_Ar_2mega_NUISPREP_flat.root
  KEY: TTree    FlatTree_VARS;9 FlatTree_VARS
  KEY: TTree    FlatTree_VARS;8 FlatTree_VARS
  KEY: TH1D FlatTree_FLUX;1 FlatTree_FLUX
  KEY: TH1D FlatTree_EVT;1  FlatTree_EVT
root [2] FlatTree_VARS->Print()
******************************************************************************
*Tree    :FlatTree_VARS: FlatTree_VARS                                       *
*Entries :  2000000 : Total =       517144339 bytes  File  Size =  259478841 *
*        :          : Tree compression factor =   1.99                       *
******************************************************************************
*Br    0 :Mode      : Mode/I                                                  *
*Entries :  2000000 : Total  Size=    8003990 bytes  File Size  =    1296998 *
*Baskets :       37 : Basket Size=     933376 bytes  Compression=   6.17     *
*.............................................................................*
```

This is the flat-tree

Looking inside the trees

- Explore these at your leisure; have things like q_3, number of final state particles, momentum along x, y, z, signal flags...

```
*.....................................................................*
*Br   12 :q3         : q3/F                                           *
*Entries :  2000000 : Total  Size=    8003908 bytes  File Size  =    7122877 *
*Baskets :       37 : Basket Size=     933376 bytes  Compression=   1.12     *
*.....................................................................*
*Br   26 :nfsp       : nfsp/I                                         *
*Entries :  2000000 : Total  Size=    8003990 bytes  File Size  =    1379732 *
*Baskets :       37 : Basket Size=     933376 bytes  Compression=   5.80     *
*.....................................................................*
*Br   27 :px         : px[nfsp]/F                                     *
*Entries :  2000000 : Total  Size=   36192505 bytes  File Size  =   29748691 *
*Baskets :      176 : Basket Size=    4230144 bytes  Compression=   1.22     *
*.....................................................................*
*Br   41 :flagCC0pi : flagCC0pi/O                                     *
*Entries :  2000000 : Total  Size=    2001984 bytes  File Size  =     422722 *
*Baskets :       16 : Basket Size=     233472 bytes  Compression=   4.73     *
*.....................................................................*
```

Mode listings

- Mode entry follows NEUT mode; exists for NEUT, GENIE and NuWro using the official conversions

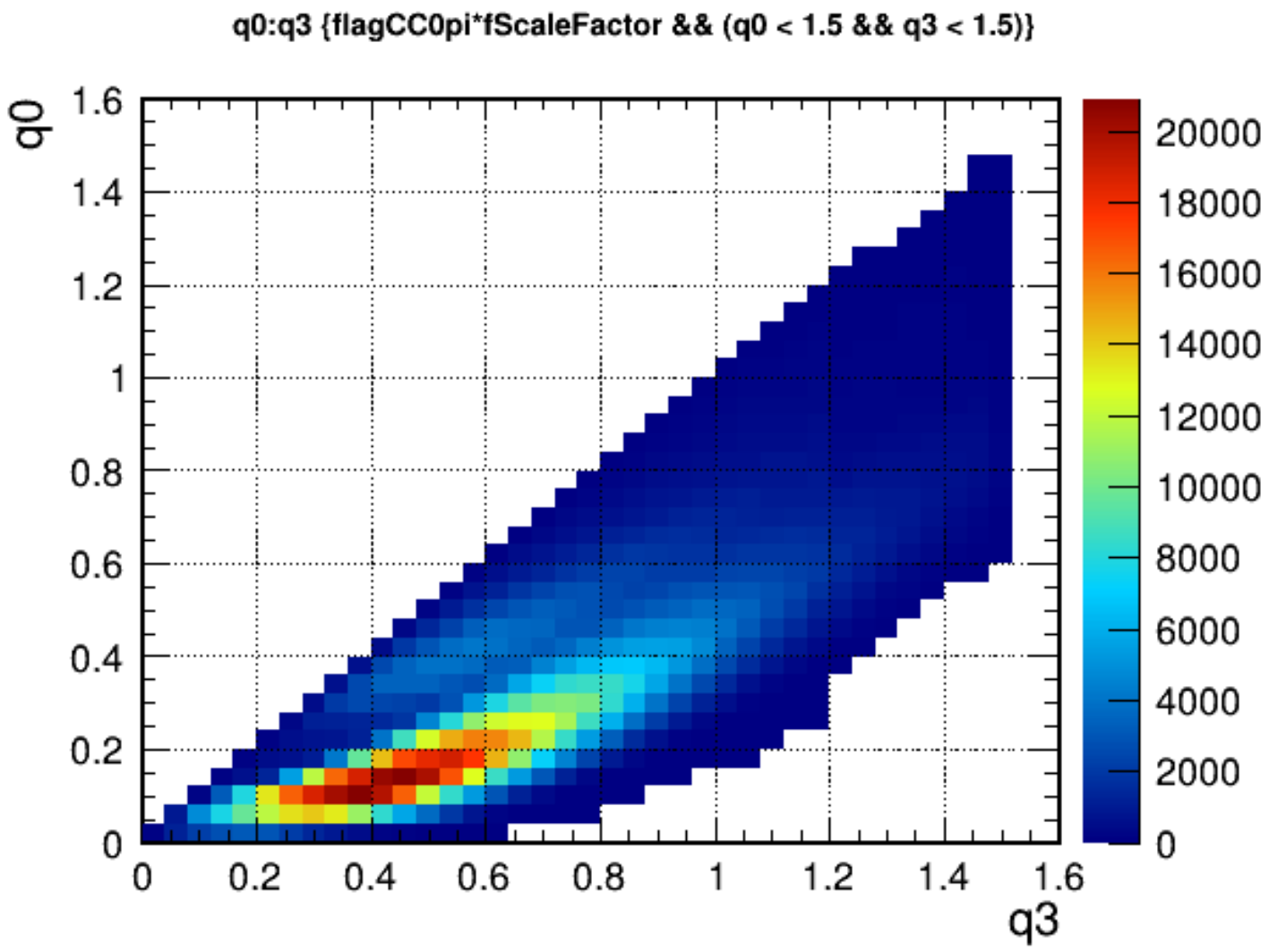
1	CC QE
2	2p2h
11	CC $1\pi^+1p$
12	CC $1\pi^0$
13	CC $1\pi^+1n$
16	CC coh
17	CC 1γ
21	CC Multi-π
22	CC 1η
23	CC 1K
26	CC DIS

31	NC $1\pi^0 1n$
32	NC $1\pi^0 1p$
33	NC $1\pi^- 1p$
34	NC $1\pi^+ 1n$
36	NC coh
38	NC 1γ 1n
39	NC 1γ 1p
41	NC multi-π
42	NC 1η 1n
43	NC 1η 1p
44	NC $1K^0$ 1Λ
45	NC $1K^+$ 1Λ
46	NC DIS
51	NC QE 1p
52	NC QE 1n

Simple draws

- Plot event distributions for CC0π events

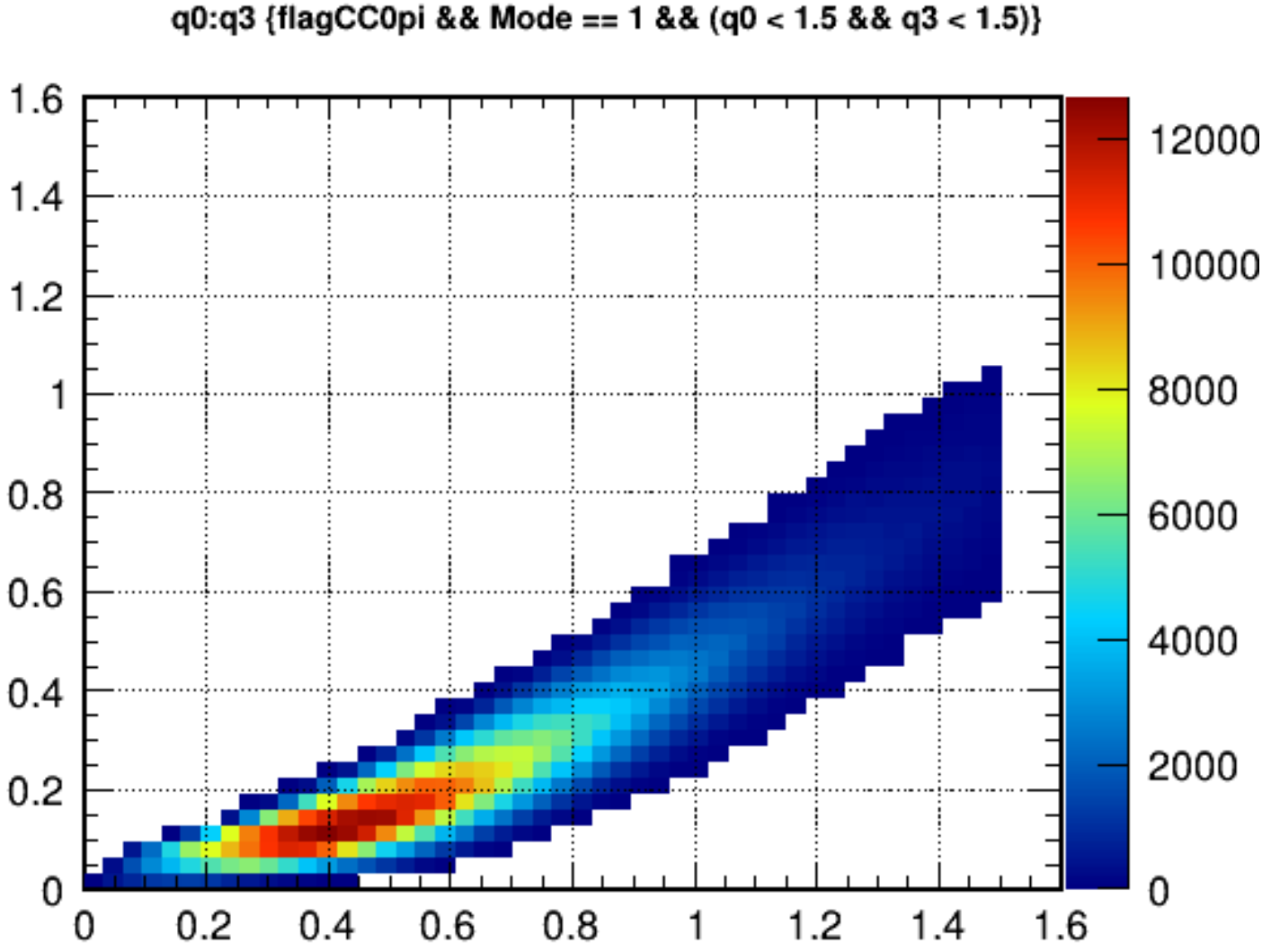
```
root [9] FlatTree_VARS->Draw("q0:q3", "flagCC0pi && (q0 < 1.5 && q3 < 1.5)", "colz")
```



Simple draws

- Plot event distributions for CC0π events, for true CCQE

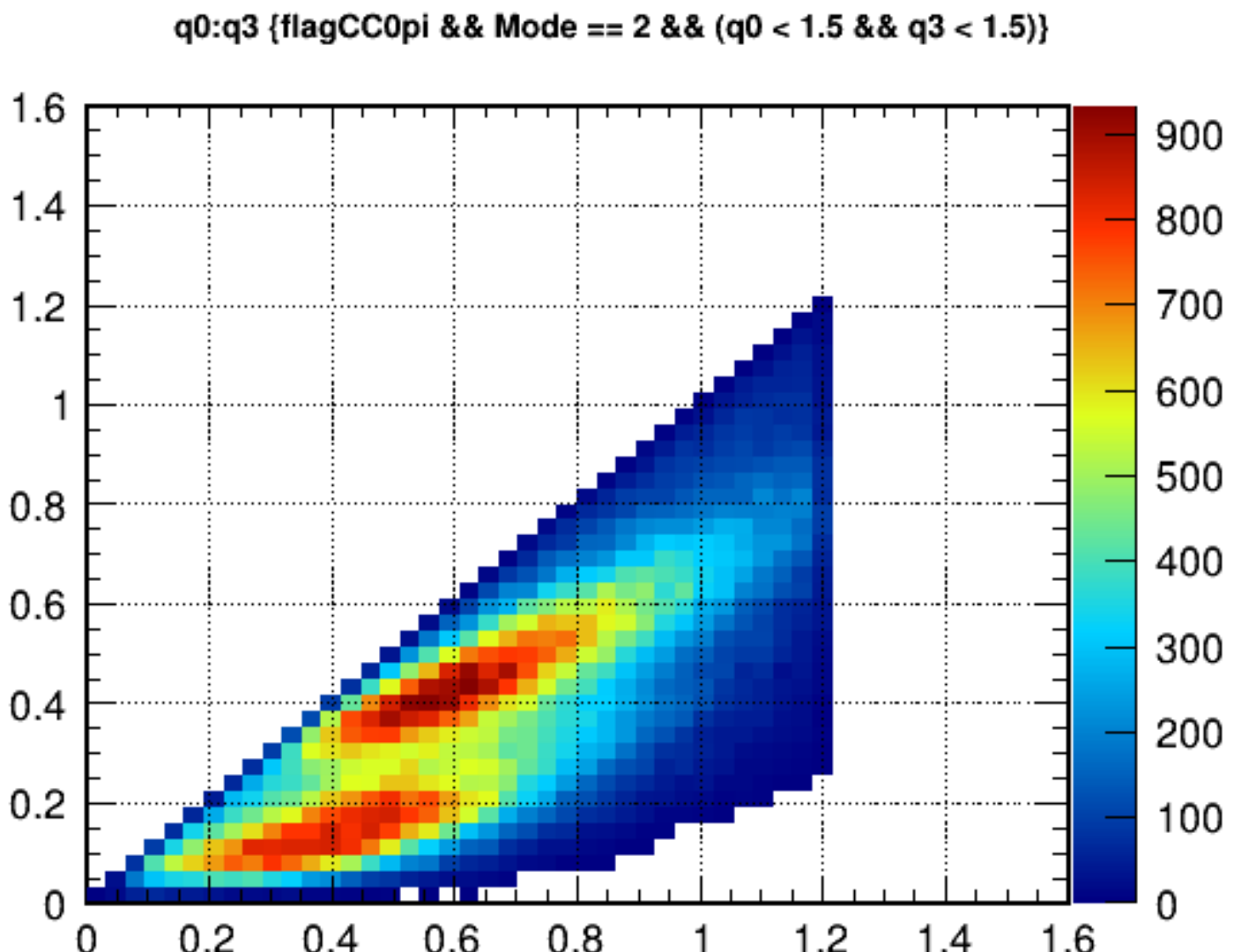
```
root [9] FlatTree_VARS->Draw("q0:q3", "flagCC0pi && Mode == 1 && (q0 < 1.5 && q3 < 1.5)", "colz")
```



Simple draws

- Plot event distributions for CC0π events, for true 2p2h

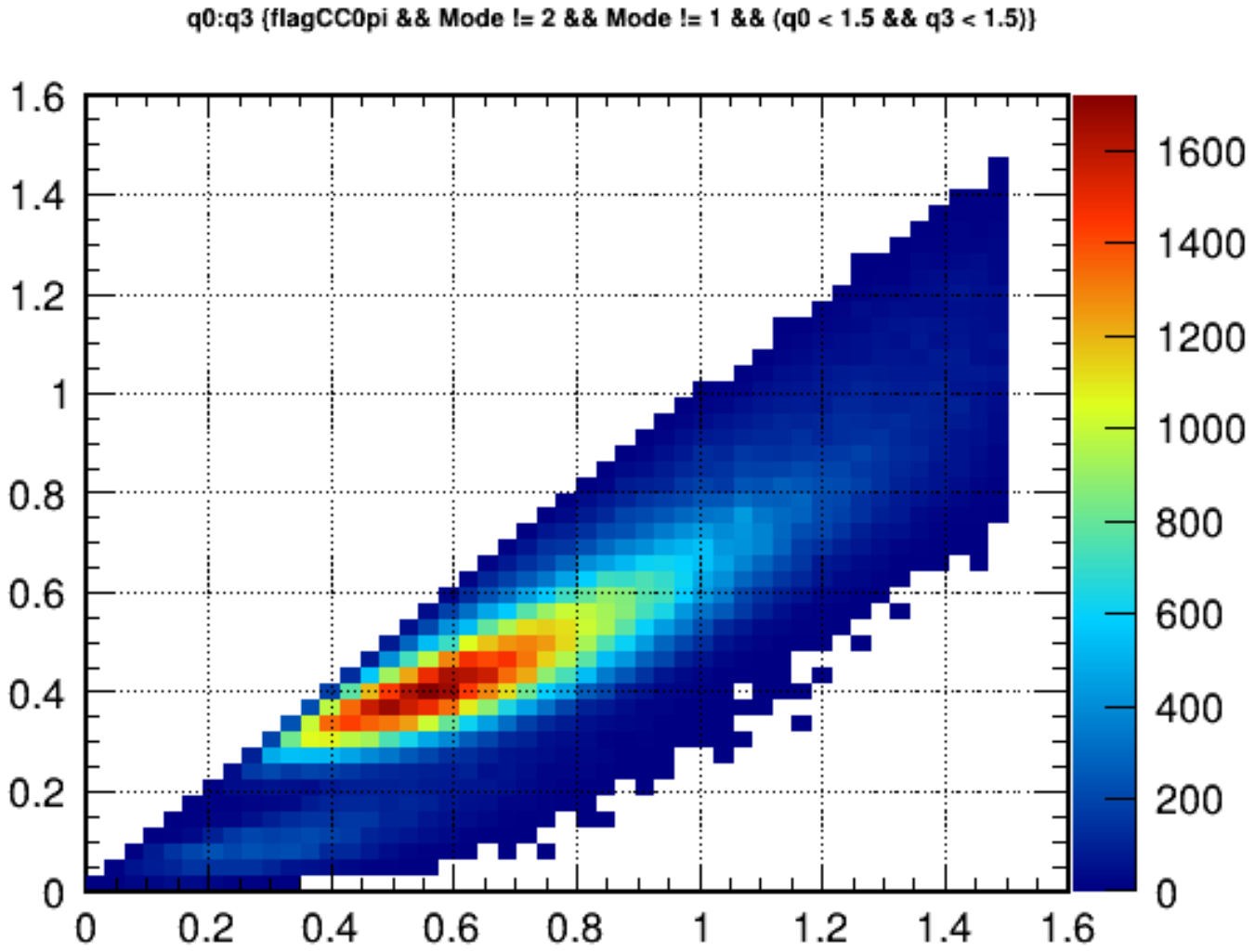
```
root [9] FlatTree_VARS->Draw("q0:q3", "flagCC0pi && Mode == 2 && (q0 < 1.5 && q3 < 1.5)", "colz")
```



Simple draws

- Plot event distributions for CC0π events, for anything not CCQE or 2p2h

```
root [9] FlatTree_VARS->Draw("q0:q3", "flagCC0pi && Mode != 1 && Mode != 2 && (q0 < 1.5 && q3 < 1.5)", "colz")
```



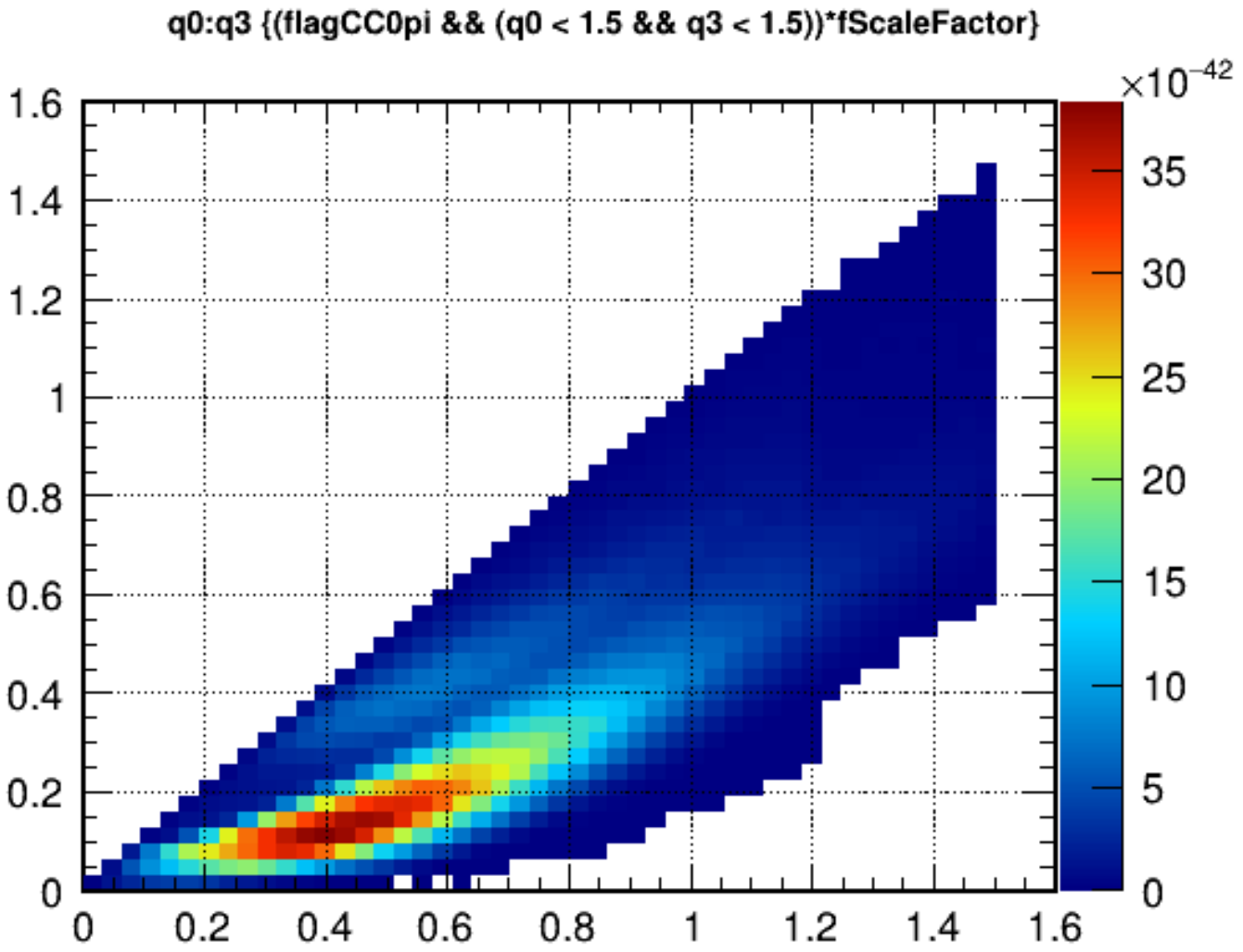
Scaling to a cross-section

- NuWro, GENIE, NEUT all generate events in proporation to xsec (GiBUU does not!)
- `fScaleFactor` takes you from event rate to cross-section: the same for all events for these generators
- Pre-calculated for you, just needs applying to your distribution
- **Beware**: `fScaleFactor` is often very small (~1E-45), so scaling a `TH1F` (with floats) will produce empty plots
  - `TTree->Draw` often writes to a `TH1F`
  - **Might miss entries!**

Simple draws with xsec

- Plot event distributions for CC0π events, now with xsec

```
root [9] FlatTree_VARS->Draw("q0:q3", "(flagCC0pi && (q0 < 1.5 && q3 < 1.5))*fScaleFactor", "colz")
```

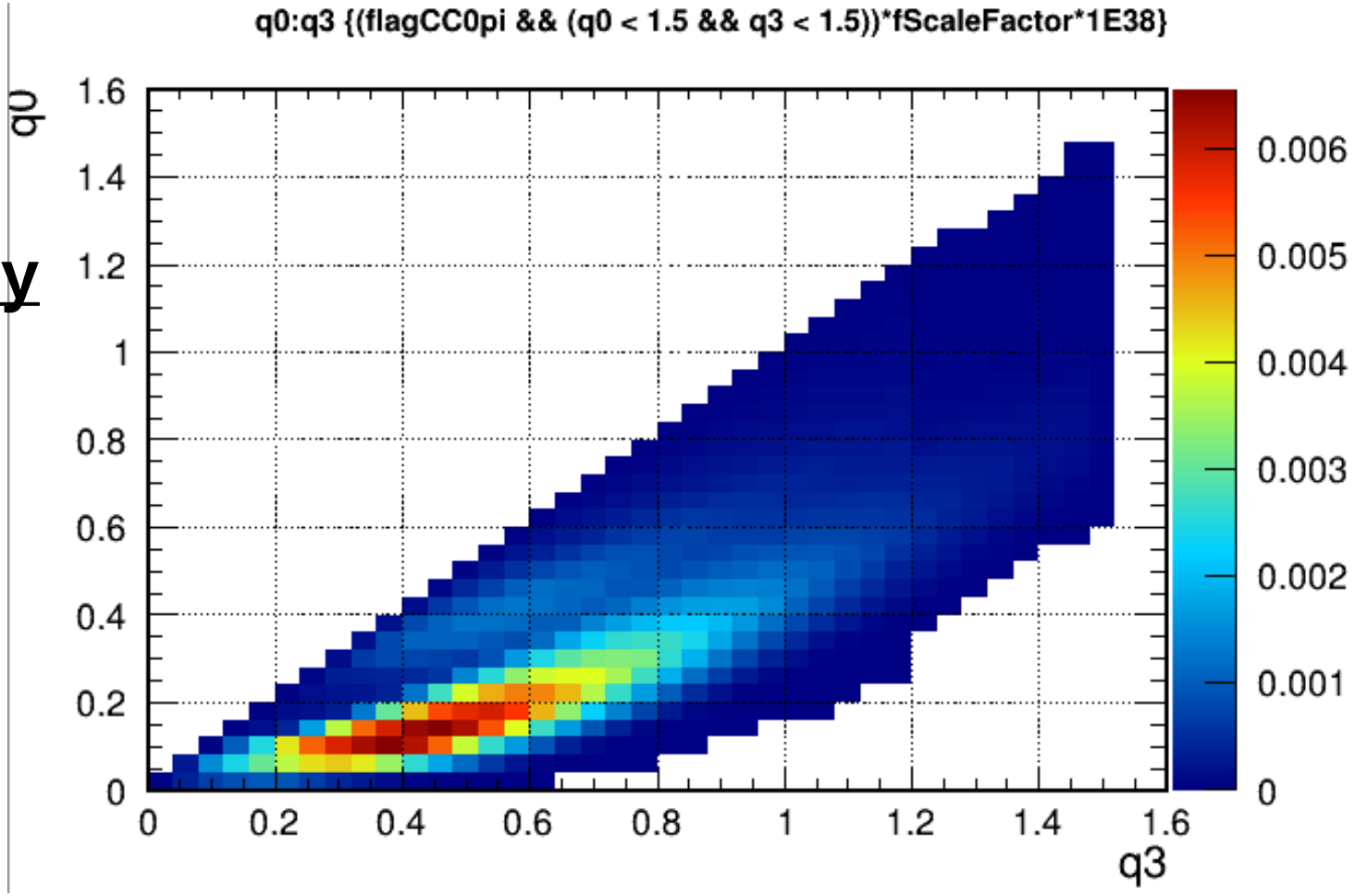


Simple draws with xsec

- Plot event distributions for CC0π events, now with xsec and fixed scaling

```
root [9] FlatTree_VARS->Draw("q0:q3", "(flagCC0pi && (q0 < 1.5 && q3 < 1.5))*fScaleFactor*1E38", "colz")
```

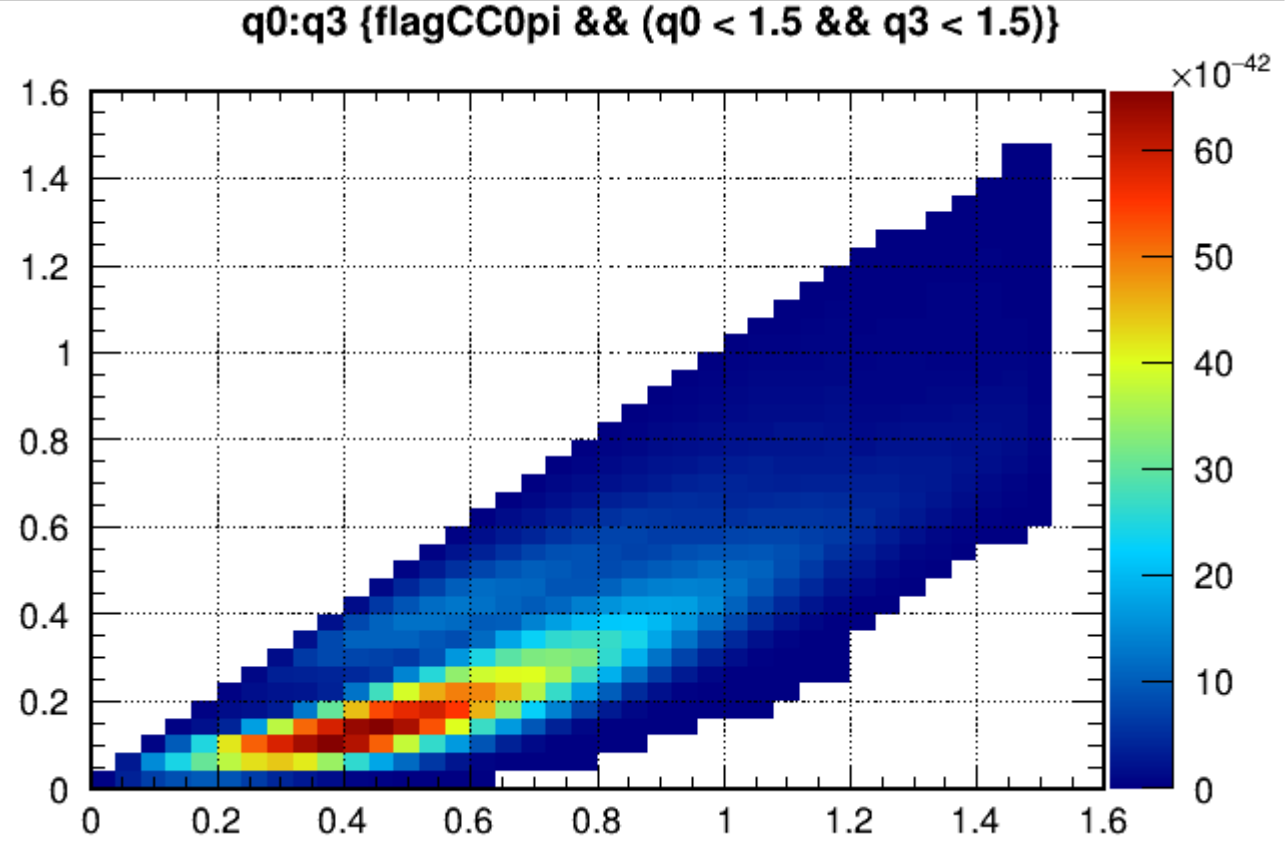
Indeed the early `TTree::Draw` missed a lot of xsec



Simple draws with xsec

- Or using different method

```
root [1] FlatTree_VARS->Draw("q0:q3>>h1", "flagCC0pi && (q0 < 1.5 && q3 < 1.5)", "colz")
root [2] TH2D *h1 = (TH2D*)gDirectory->Get("h1")->Clone()
root [3] double scale = FlatTree_VARS->GetMaximum("fScaleFactor")
root [4] h1→Scale(scale)
root [5] h1->Draw("colz")
```



Indeed the early `TTree::Draw` missed a lot of xsec

Comparing some generators

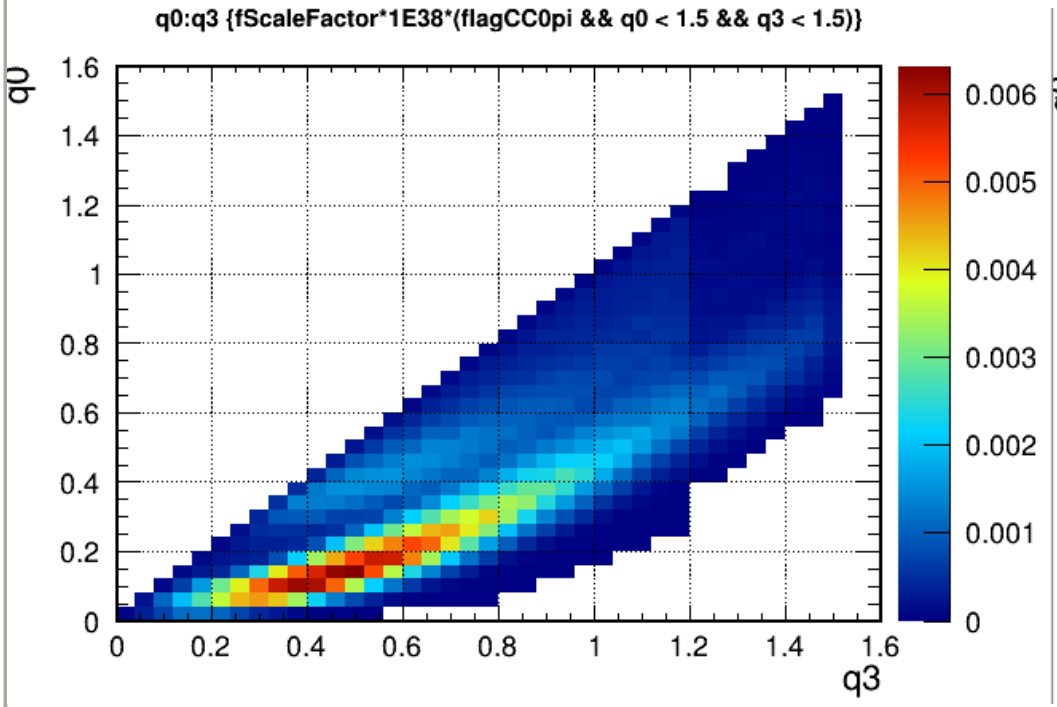
- Now draw different generators to your hearts content!

```
root [1] TFile *nuwro = new Tfile("nuwro_minerva_numu_CH_2mega_NUISPREP_flat.root",
"open")
root [2] TFile *genie = new Tfile("tensions-2019-MINERvA-numu-G18_02a_02_11a-
NUISANCE.flat.root", "open")
root [3] nuwro->cd()
root [4] FlatTree_VARS->Draw("q0:q3", "fScaleFactor*1E38*(flagCC0pi && q0 < 1.5 &&
q3 < 1.5)","colz")
root [7] genie->cd()
root [8] FlatTree_VARS->Draw("q0:q3", "fScaleFactor*1E38*(flagCC0pi && q0 < 1.5 &&
q3 < 1.5)","colz")
```

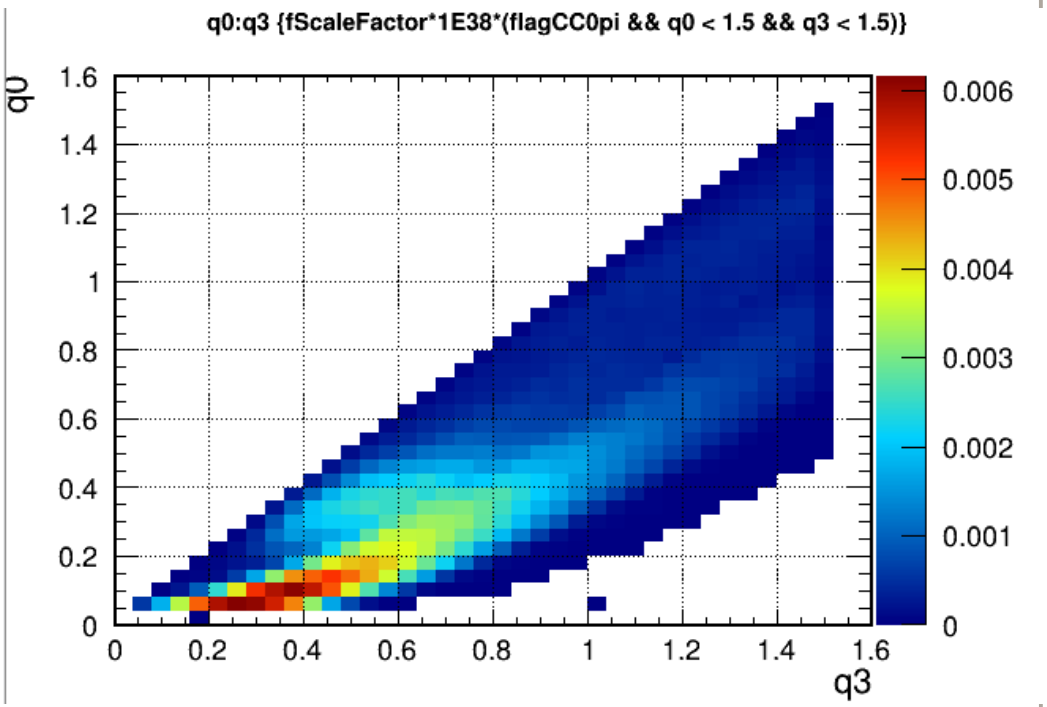
Draws the CC0pi prediction for MINERvA for NuWro and GENIE with Empirical MEC

Comparing some generators

NuWro



GENIE with Empirical MEC



More fun draws with xsec

- Have access to full particle stack
- Maybe I want to count the number of particles above/below a threshold for a selection, comparing the generators
- Code available at

```
/pnfs/dune/persistent/users/cwret/Pittsburgh/flattened/flatme.cpp
```

- And on the box

More fun draws with xsec

```
void flatme(std::string filename) {
  TFile *file = new TFile(filename.c_str(), "open");
  TTree *tree = (TTree*)file->Get("FlatTree_VARS");
  int nentries = tree->GetEntries();
  const int nmax = 100;
  bool flagCC1pip;
  float E[nmax];
  float px[nmax];
  float py[nmax];
  float pz[nmax];
  int pdg[nmax];
  int Mode;
  int nfsp;
  tree->SetBranchStatus("*", false);
  tree->SetBranchStatus("flagCC1pip", true);
  tree->SetBranchAddress("flagCC1pip", &flagCC1pip);
  tree->SetBranchStatus("E", true);
  tree->SetBranchAddress("E", &E);
  tree->SetBranchStatus("px", true);
  tree->SetBranchAddress("px", &px);
  tree->SetBranchStatus("py", true);
  tree->SetBranchAddress("py", &py);
```

Just setting things up as usual when doing TTree analyses

Please ask questions if unclear!

More fun draws with xsec

```
tree->SetBranchStatus("pz", true);
  tree->SetBranchAddress("pz", &pz);
  tree->SetBranchStatus("pdg", true);
  tree->SetBranchAddress("pdg", &pdg);
  tree->SetBranchStatus("Mode", true);
  tree->SetBranchAddress("Mode", &Mode);
  tree->SetBranchStatus("nfsp", true);
  tree->SetBranchAddress("nfsp", &nfsp);
  tree->SetBranchStatus("fScaleFactor", true);
  double scale = tree->GetMaximum("fScaleFactor");
  const double picut = 0.2;
  TH1D *ppi = new TH1D("ppi", "ppi", 100, 0, picut);
  TFile *out = new TFile(Form("%s_out.root", filename.c_str()), "recreate");
```

More setting up

Getting the scale factor

Setting the cut

The histogram we're filling

The output file

More fun draws with xsec

```
  for (int i = 0; i < nentries; ++i) {
    tree->GetEntry(i);

    if (i % (nentries/5) == 0) std::cout << i << "/" << nentries << " (" <<
(double(i)/nentries)*100 << "%)" << std::endl;
    if (!flagCC1pip) continue;
    int nPions = 0;
    int PiIndex = 0;
    for (int j = 0; j < nfsp; ++j) {
      if (pdg[j] != 211) continue;
      nPions++;
      PiIndex = j;
    }
    if (nPions != 1) continue;
    double ptot = sqrt(px[PiIndex]*px[PiIndex]+py[PiIndex]*py[PiIndex]
+pz[PiIndex]*pz[PiIndex]);
    if (ptot > picut) continue;
    ppi->Fill(ptot);
  }
```

The actual event loop

Get the entry

Signal definition

Count pions

Loop over the final state particles

Check PDG of final state particle j

Count pions

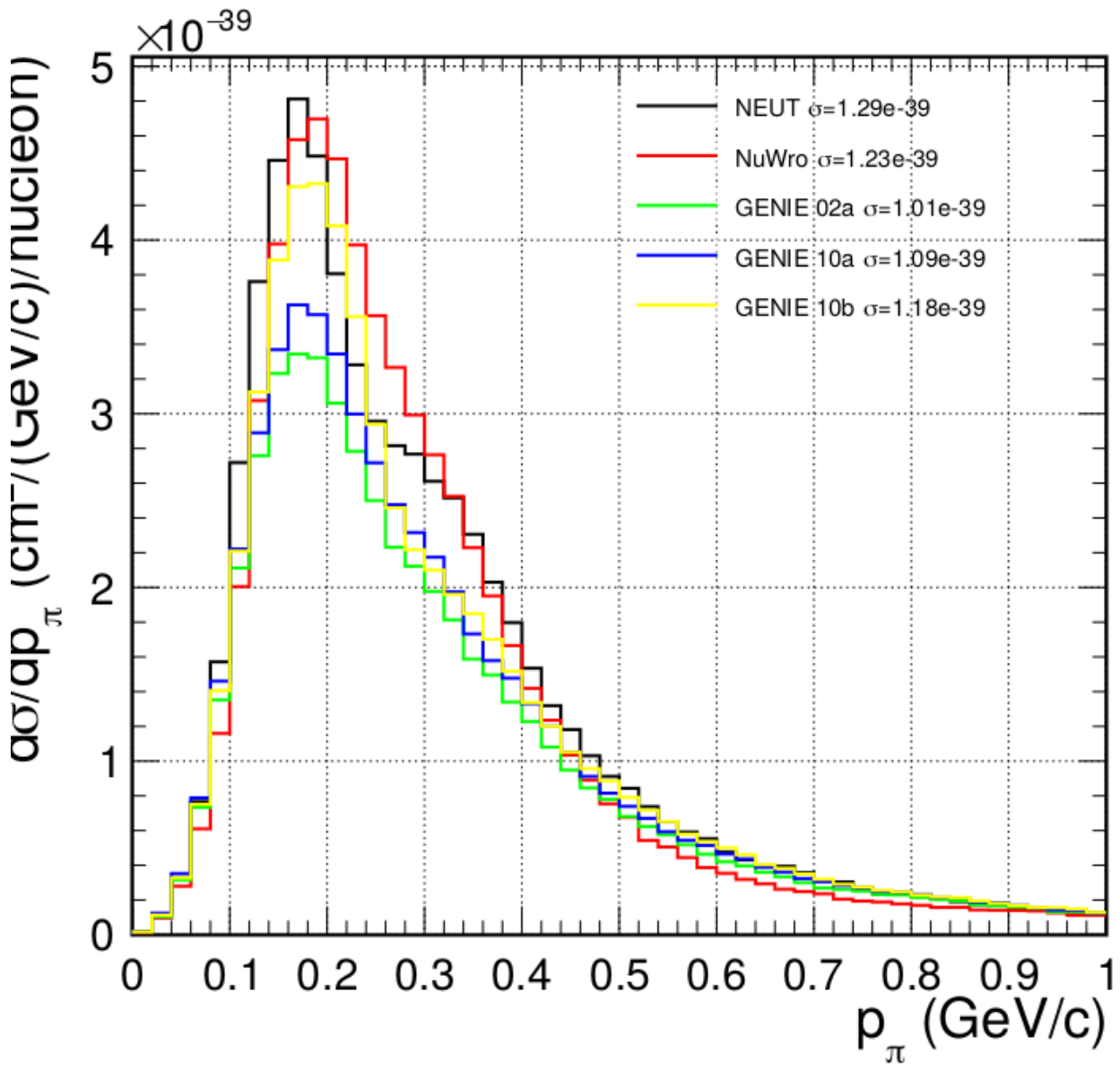
Look for one pion

Calculate momentum

If too high momentum, move on

More fun draws with xsec

- Run over all the flat-trees and plot them (here without the cut!)



Questions?

Let’s hack!