ProtoDUNE analysis workshop
CERN, 26/01/2020

The HighLAND analysis framework

Anselmo Cervera Villanueva
IFIC-Valencia

Outline

- Introduction
- HighLAND in few hours
- HighLAND concepts
- Event selection
- Systematic error propagation

HighLAND analysis framework

- **HighLAND: High L**evel **An**alysis **D**evelopment
- HighLAND has been crucial for T2K near detector analyses
- **Highly optimized, thread safe, compiled c++ code** and run on the shell command line (not as root macro)
- **Very compact set of packages**: 1 minute to download and 5 minutes to compile
- **Functionality**:
  - Event selection & systematics propagation
  - Drawing Tools & Event display
  - Data reduction

We have not started from scratch: All this functionality exists since long time from T2K. The system is fully validated !!!!

Previous HighLAND talks

- **MCC6** and **MCC7** eras
  - **FD sim/reco 23/11/2015**: https://indico.fnal.gov/conferenceDisplay.py?confId=10882
  - **LBL 24/11/2015**: https://indico.fnal.gov/conferenceDisplay.py?confId=10861
  - **S&C 15/12/2015**: https://indico.fnal.gov/conferenceDisplay.py?confId=11030
  - **DUNE CM, 14/09/2016**, https://indico.fnal.gov/event/10613/session/18/contribution/52/material/slides/0.pdf
  - **PD meas/ana 13/10/2016**: https://indico.fnal.gov/event/13110/
  - **DUNE CM 24/01/2017**: https://indico.fnal.gov/event/10641/session/12/contribution/81/material/slides/0.pdf
- **MCC11, MCC12** era
  - ProtoDUNE analysis workshop 27/01/2019
    - https://indico.fnal.gov/event/19133/
    - Long talk explaining the framework and its functionality
  - DRA Analysis meeting 9/05/2019
    - https://indico.fnal.gov/event/20776/
    - Actual ProtoDUNE analyses using HighLAND
  - DUNE CM, 21/05/2019
    - https://indico.fnal.gov/event/18681/session/13/contribution/93/material/slides/0.pdf
    - Actual ProtoDUNE analyses using HighLAND
  - Analysis meeting 5/12/2019
    - https://indico.fnal.gov/event/22522/
    - HighLAND as candidate for systematic error propagation

HighLAND in few hours

HighLAND in few hours

- This is the HighLAND redmine wiki page
  https://cdcvs.fnal.gov/redmine/projects/highland/wiki

link to detailed doxygen documentation

Overview Activity Issues Spent time Gantt Calendar News Documents Wiki Files Repository

The HighLAND analysis framework

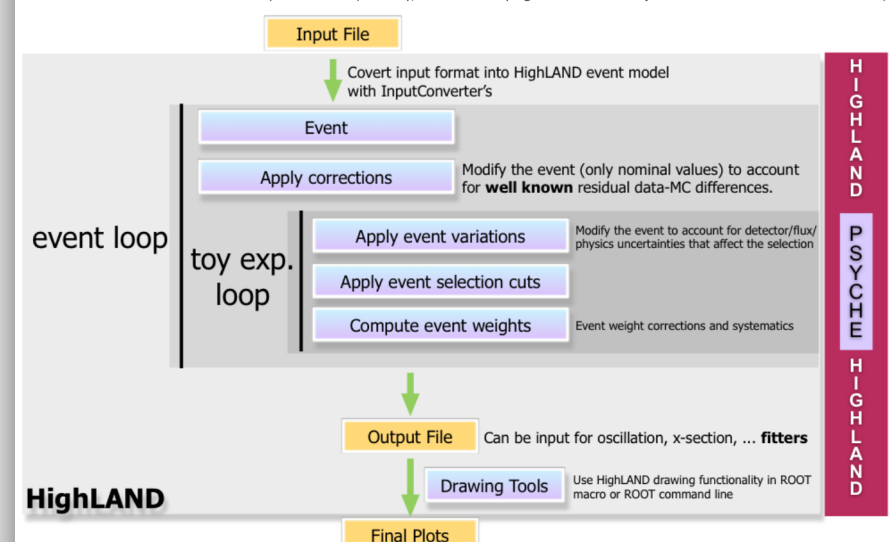
History

The full documentation generated by Doxygen can be found at http://www.cern.ch/acervera/highland. Here only a summary is shown. Also information on how to download the framework for DUNE is found at the bottom of this page.

The HighLAND framework (HIGH Level Analysis at a Neutrino Detector), initially designed to simplify the process of analysing data in the T2K near detector (ND280) is being adapted for DUNE. It allows the user to quickly run analyses, plot results, evaluate the impact of systematic errors and much more. To allow this two set of packages have been identified:

- **PSyChE** (Propagation of SYstematics and CHaracterization of Events) is a high performance set of packages which performs event selection and propagation of systematics.
- **HighLAND**: use the PSyChE packages as core, and it adds in addition Corrections, FlatTree creation, tools for Drawing, etc. In practice PSyChE is part of the HighLAND distribution. The main reason to keep them separated is that PSyChE cab be run directly by fitters.

The code if **self-contained** and has a unique external dependency, ROOT. It is very **light** and can be **compiled in less than 5 minutes** in most operating systems, being specially interesting for laptops.



The analysis flow is in the above diagram. For every event a set of corrections are applied to make data and MC agree better. Then a loop over toy experiments is done, each toy experiment will be an independent analysis of the same event, in which the event properties are varied. This allows propagating systematic errors numerically. For each toy experiment, a set of event variations change some aspect of the data. The event selection proceeds then on that modified data. Finally, a weight for the event in this particular toy experiment is computed, to account for systematics that cannot be propagated as variations, and for event weight corrections. Those three operations, event variations, event selection and event weights are performed by PSyCHE. HighLAND provides in addition the infraestructure to do the event loop, apply corrections, create and fill root trees that can be easily analysed using a set of tools for drawing (also provided by HighLAND), extensions to the event model, etc.

Each analysis that uses the framework defines its own executable, for example RunDuneExampleAnalysis.cxx for the duneExampleAnalysis package. The output of this executable is a root file with several trees. Some of them contain a summary of the variables used in the analysis as well as selection cuts information (those are called micro-trees: default, all_syst, ...), while there are also single-entry trees containing more general information such as the amount of POT that was analysed (header tree), and configuration information for the analysis (config tree). The output file can then be analysed using a simple ROOT macro, and the DrawingTools methods provide many useful functions for doing this.

The following related pages contain more detailed documentation:

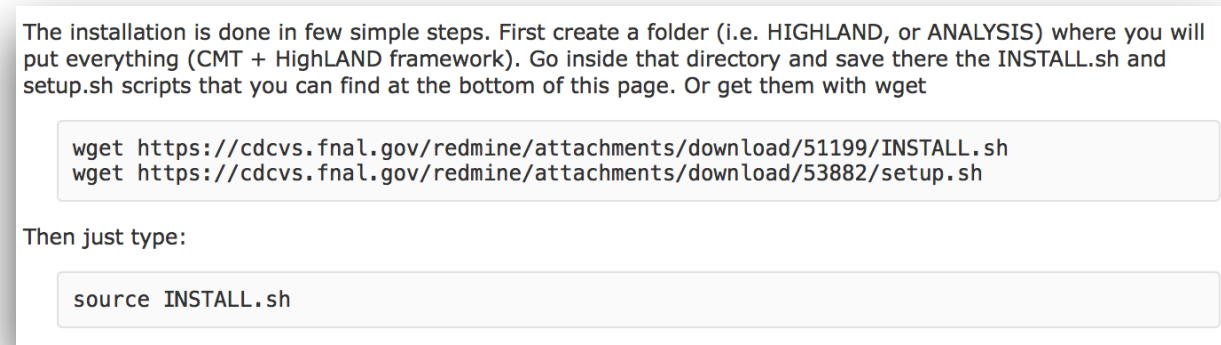
- **Download and install the HighLAND framework**: install
- **Run the DUNE example**: example
- **Run the ProtoDUNE example**: protoDUNEexample

instructions and scripts to install and run →

Results in 10 minutes

- Download, compile and install the framework

https://cdcvs.fnal.gov/redmine/projects/highland/wiki/Install



The installation is done in few simple steps. First create a folder (i.e. HIGHLAND, or ANALYSIS) where you will put everything (CMT + HighLAND framework). Go inside that directory and save there the INSTALL.sh and setup.sh scripts that you can find at the bottom of this page. Or get them with wget

```
wget https://cdcvs.fnal.gov/redmine/attachments/download/51199/INSTALL.sh
wget https://cdcvs.fnal.gov/redmine/attachments/download/53882/setup.sh
```

Then just type:

```
source INSTALL.sh
```

5 minutes

- Run the ProtoDUNE example:

```
../Linux-x86_64/RunProtoDuneExampleAnalysis.exe -n 10000 -v -o output.root input.root
```

- Where input.root can be a LArSoft reco file or a HighLAND minitree, for example:

```
/dune/data/users/acervera/MiniTreesFilter/mini_data_run5387_calocorr_filter_pos_tof0-250.root
```

1 minute for 10K events

Make a plot

```
root -l output.root
root [1] DrawingToolsBase draw("output.root")
root [2] TTree* d   = (TTree*)_file0->Get("default");

root [3] draw.Draw(d,"beam_mom_raw",100,0,2,"beamparticle","accum_level[0][0]>2")
```

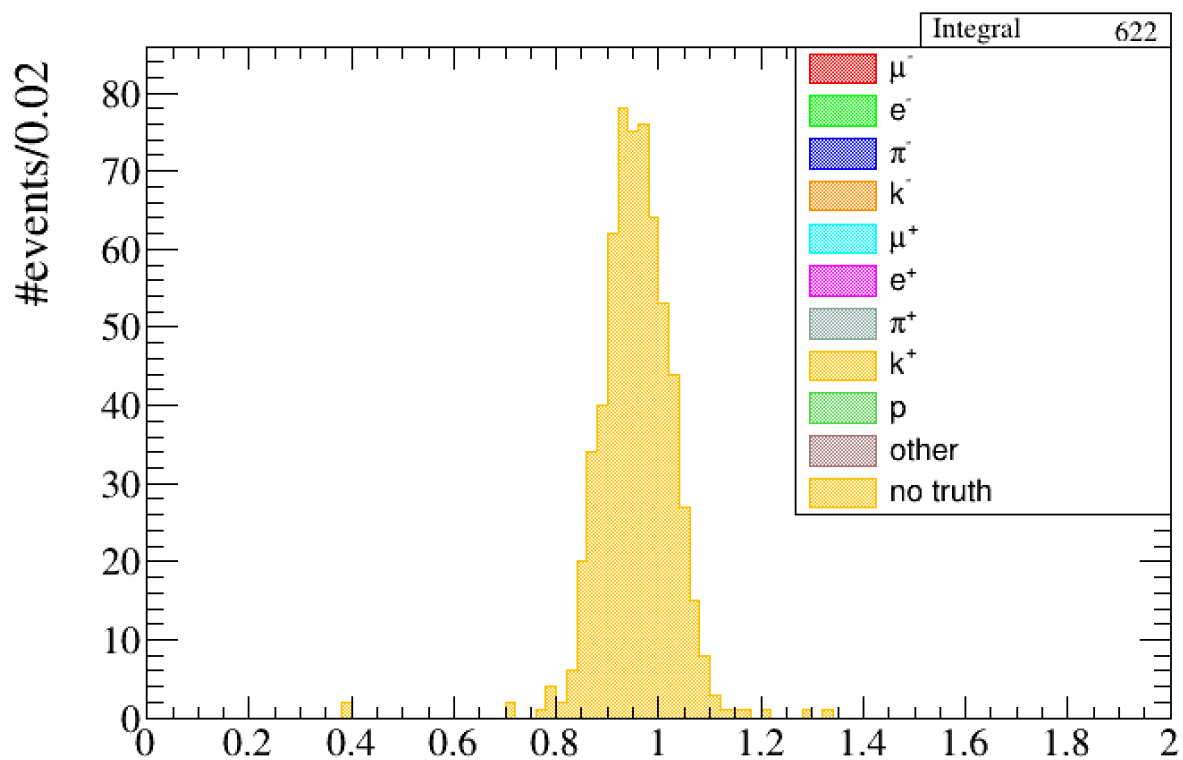
root tree

variable to plot

binning

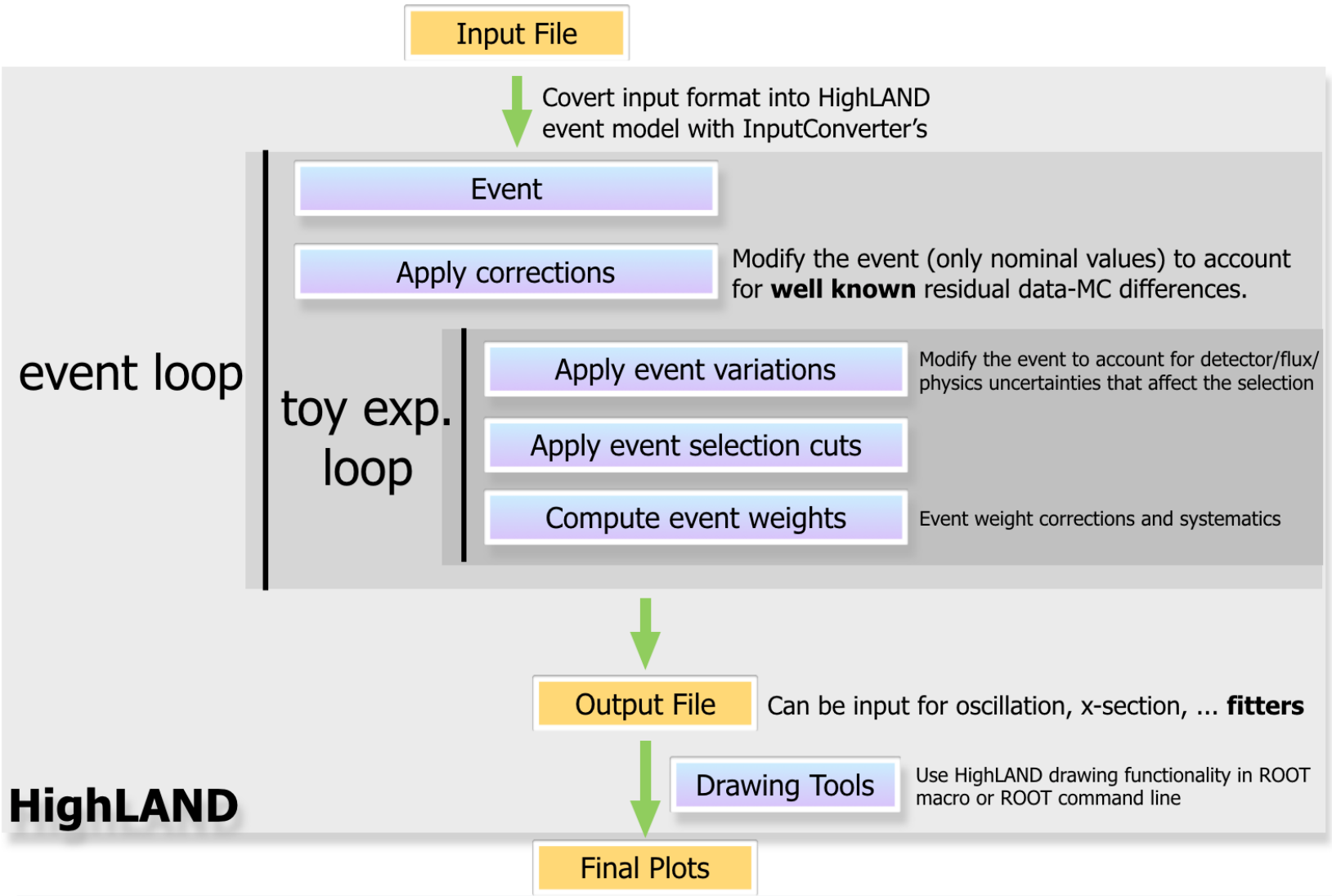
color categories

pass cut 2

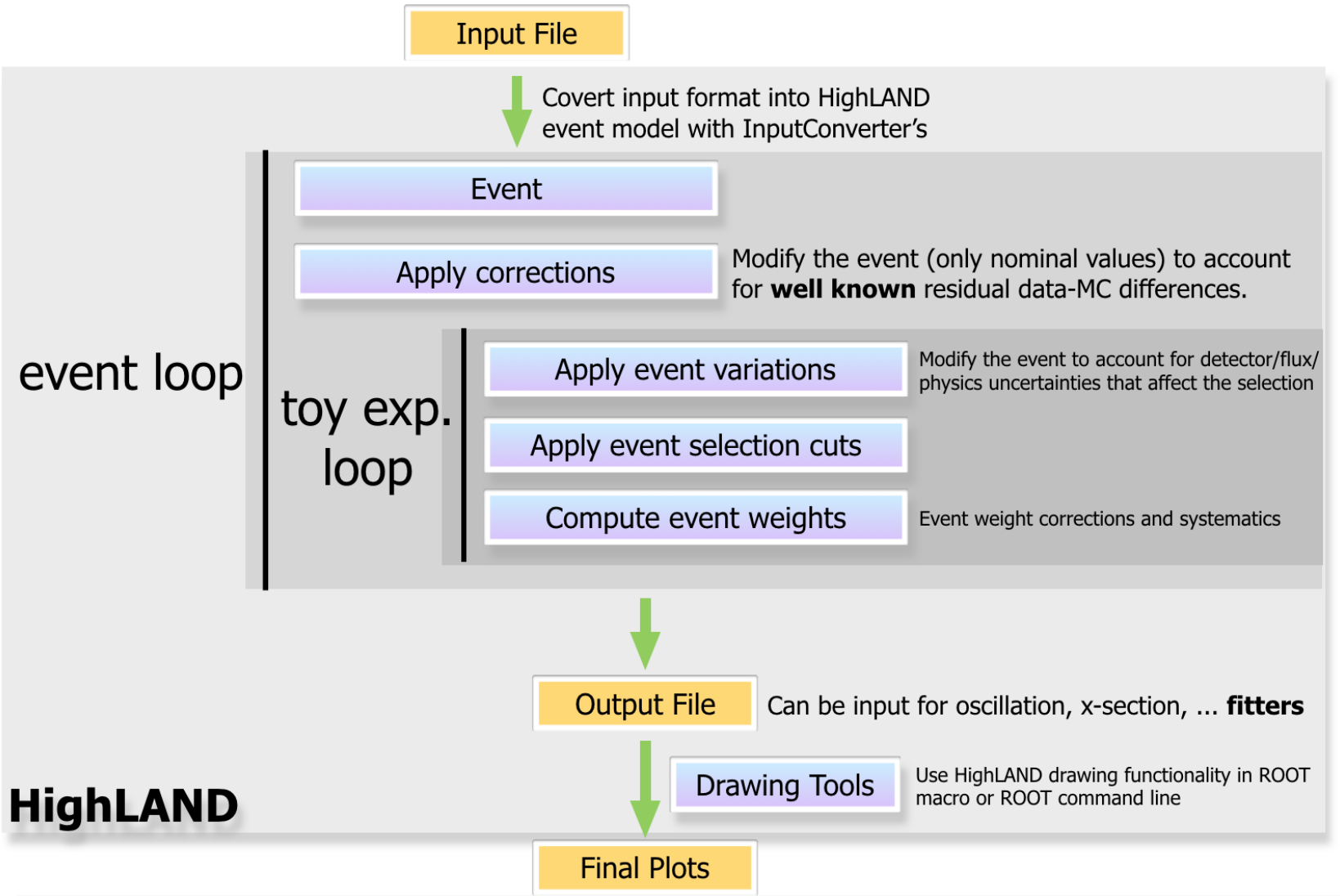


Highland concepts

Highland flow and concepts

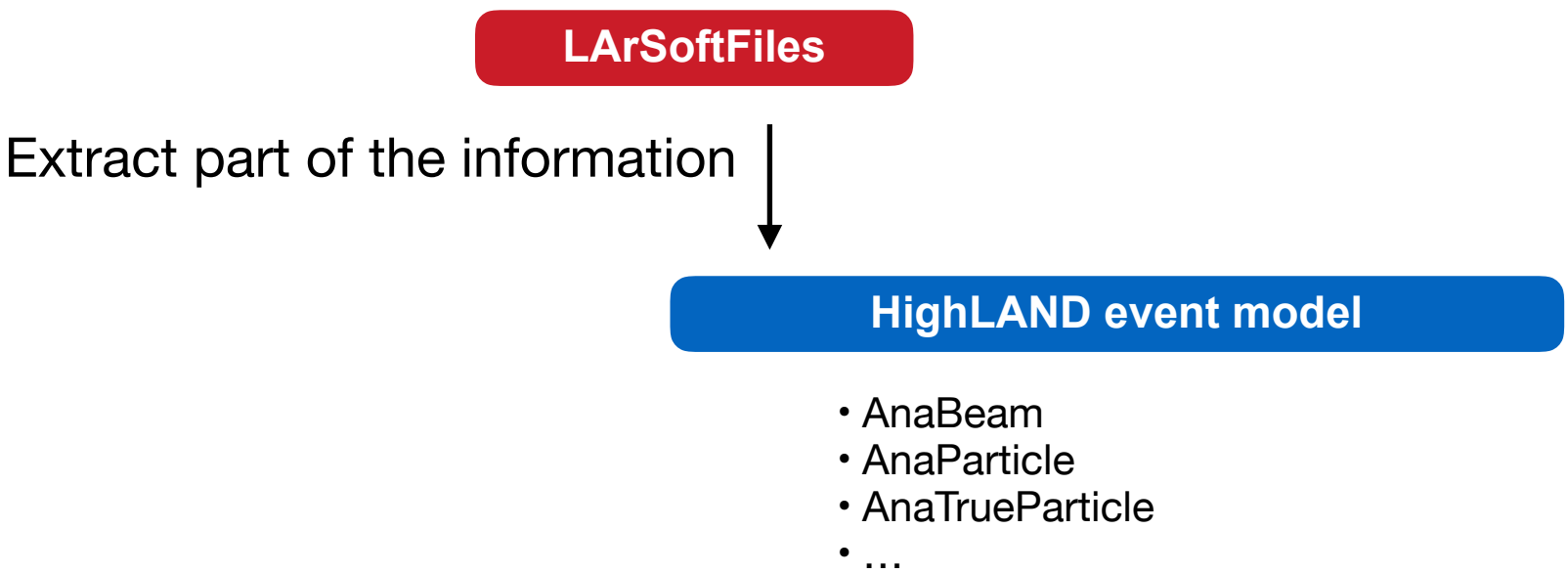


Highland flow and concepts



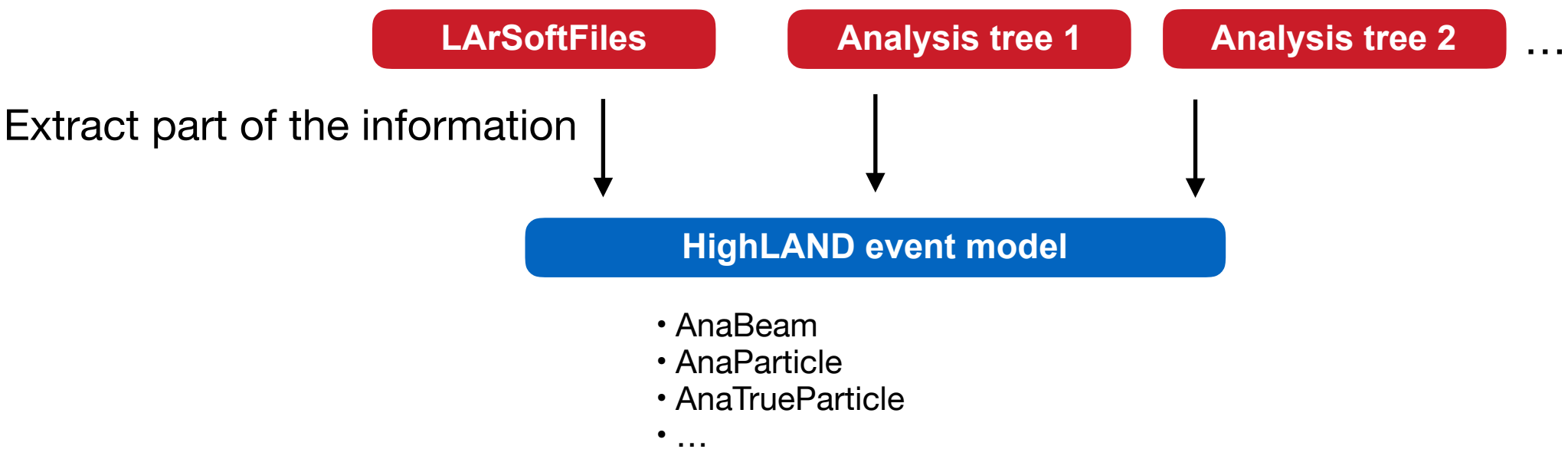
Event model

- HighLAND decouples input file's format from actual analysis by extracting info from the input file and saving it into an internal event model



- In this way the analysis is independent of the input format

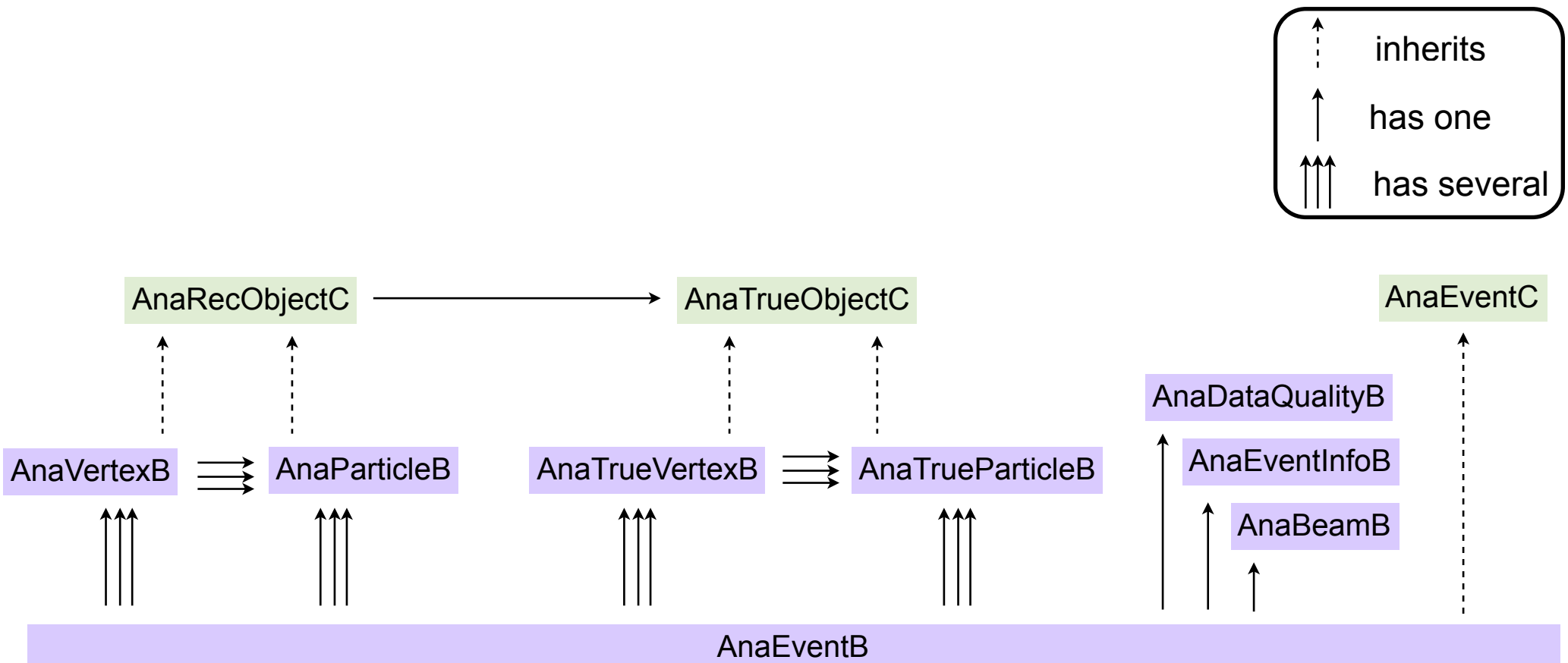
- HighLAND decouples input file's format from actual analysis by extracting info from the input file and saving it into an internal event model



- In this way the analysis is independent of the input format

HighLAND event model

- The event model is very simple
- This is the current event model, which can be easily modified



Some classes

AnaTrueParticle

```
/// The PDG code of this particle.
Int_t PDG;

/// The ID of this particle's immediate parent, or 0
Int_t ParentID;

/// The PDG code of this particle's immediate parent,
Int_t ParentPDG;

/// The PDG code of this particle's grandparent, or 0
Int_t GParentPDG;

/// Process generating this particle
ProcessEnum ProcessStart;

/// Process destroying this particle
ProcessEnum ProcessEnd;

/// The initial position of the true particle.
Float_t Position[4];

/// The end position of the true particle.
Float_t PositionEnd[4];

/// The initial direction of the true particle.
Float_t Direction[3];

/// The end direction of the true particle.
Float_t DirectionEnd[3];

/// The initial momentum of the true particle.
Float_t Momentum;

/// The final momentum of the true particle.
Float_t MomentumEnd;

/// The true charge of the particle.
Float_t Charge;
```

```
/// The particle length
Float_t Length;

/// The particle length inside the TPC
Float_t LengthInTPC;

/// The true momentum at the TPC entrance
Float_t MomentumInTPC;
```

AnaParticle

```
/// The reconstructed start direction of the particle.
Float_t DirectionStart[3];

/// The reconstructed end direction of the particle.
Float_t DirectionEnd[3];

/// The reconstructed start position of the particle.
Float_t PositionStart[4];

/// The reconstructed end position of the particle.
Float_t PositionEnd[4];
```

```
/// PID variables
Float_t PID[3][10];

Float_t PIDA[3];

/// CALO variables
Float_t CALO[3][10];

/// Momentum by range for muon and proton hypotheses
Float_t RangeMomentum[2];

/// Vector of daugthers particles
std::vector<AnaRecObjectC*> Daughters;
```

```
/// True-reco matching efficiency
Float_t TrueEff;

/// True-reco matching purity
Float_t TruePur;

/// Number of hits in each wire plane
Int_t NHitsPerPlane[3];

/// Residual range for each wire in each plane
Float_t ResidualRange[3][NMAXHITSPERPLANE];

/// dEdx for each wire in each plane
Float_t dEdx[3][NMAXHITSPERPLANE];
Float_t dEdx_corr[3][NMAXHITSPERPLANE];

/// dQdx for each wire in each plane
Float_t dQdx[3][NMAXHITSPERPLANE];
Float_t dQdx_corr[3][NMAXHITSPERPLANE];

/// dQdx for each wire in each plane
Float_t HitX[3][NMAXHITSPERPLANE];
Float_t HitY[3][NMAXHITSPERPLANE];
Float_t HitZ[3][NMAXHITSPERPLANE];

/// Average energy deposited in the detector
Float_t AveragedEdx;

/// Average chage deposited in the detector
Float_t AveragedQdx;
```

```
/// The link to the true oject that most likely generated this reconstructed object
AnaTrueObjectC* TrueObject;
```

AnaBeam

```
/// The beam particle
AnaParticleMomB* BeamParticle;

/// Other relevant beam info
int BeamTrigger;
double TOF;
int    CerenkovStatus[2];
double CerenkovTime[2];
double CerenkovPressure[2];
double BeamTrackTime;
double BeamMomentum;
double BeamMomentumInTPC;
```

Event selection

Event selection

- Once the input file info is extracted and saved into the internal event model, the event selection can proceed
- A selection is a collection of steps, which can be actions or cuts
- Each selection inherits from **SelectionBase**, which has a main mandatory method **DefineSteps**

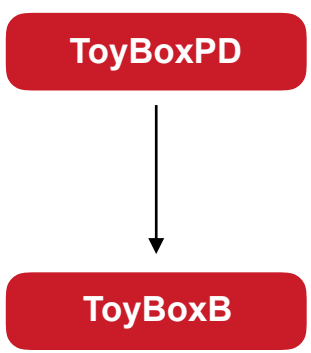
```
//********************************************************************
void stoppingProtonSelection::DefineSteps(){
//********************************************************************

  // Steps must be added in the right order
  // if "true" is added to the constructor of the step,
  // the step sequence is broken if cut is not passed (default is "false")
  AddStep(StepBase::kAction, "find main track",    new FindBeamTrackAction());
  AddStep(StepBase::kCut,    "beam protom",        new BeamProtonCut());
  AddStep(StepBase::kCut,    "beam track in TPC",  new CandidateExistsCut());
  AddStep(StepBase::kCut,    "seltrk angle cut",   new BeamProtonAngleCut());
  AddStep(StepBase::kCut,    "proton CSDA range",  new ProtonCSDARangeCut());

  SetBranchAlias(0,"trunk");
}
```

- Each step inherits from **StepBase** and implements the method **Apply**

The box

- The Toy Box is used to pass derived information from one step to another in the selection

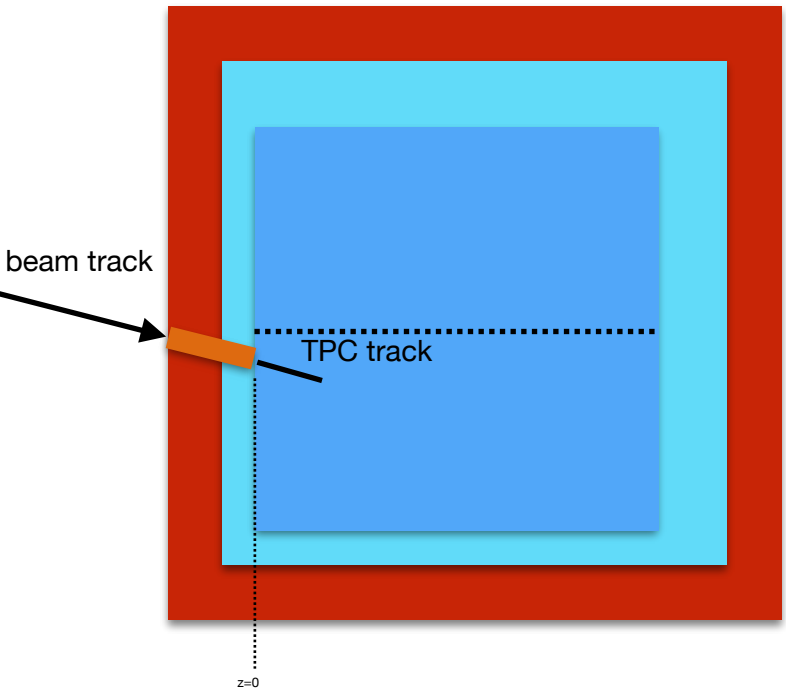
ProtoDUNE box



Base class

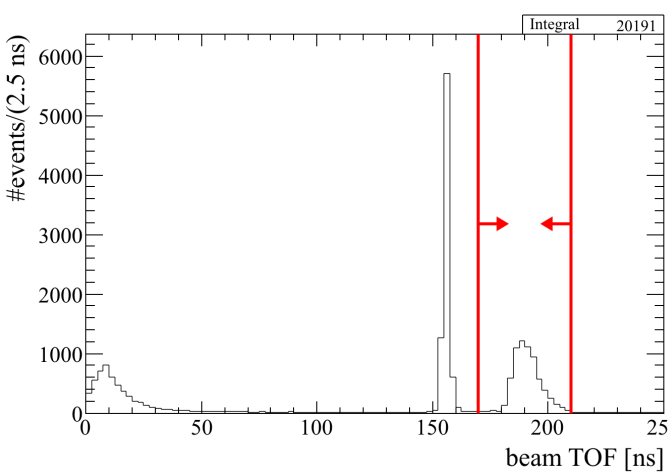
```
#ifndef ToyBoxPD_h
#define ToyBoxPD_h

#include "ToyBoxB.hxx"
#include "DataClasses.hxx"

class ToyBoxPD:public ToyBoxB{
public :

  ToyBoxPD();
  virtual ~ToyBoxPD(){}

  /// This method should be implemented by the derived class. If so it does nothing here
  virtual void Reset();

  /// Reset this base class
  virtual void ResetBase();

public:

  /// For storing the true vertex, for analyses with no reconstructed primary vertex
  AnaTrueVertexB* TrueVertex;

  /// The reconstructed EventVertex
  AnaVertexB* Vertex;

  /// The MainTrack, defining the event vertex
  AnaParticle* MainTrack;
};
```

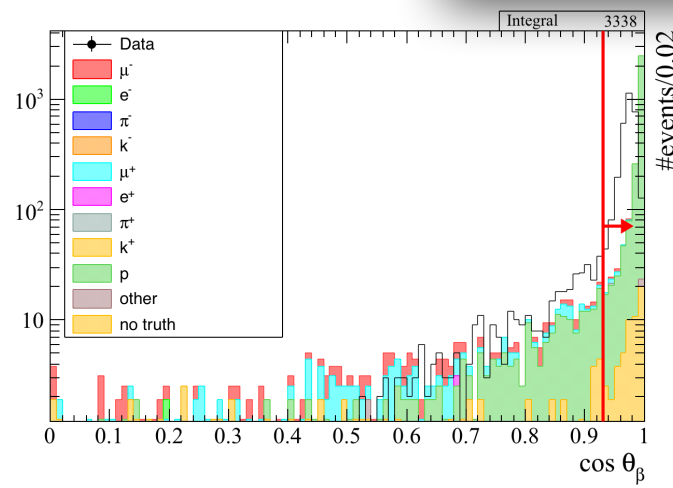
Stopping proton selection

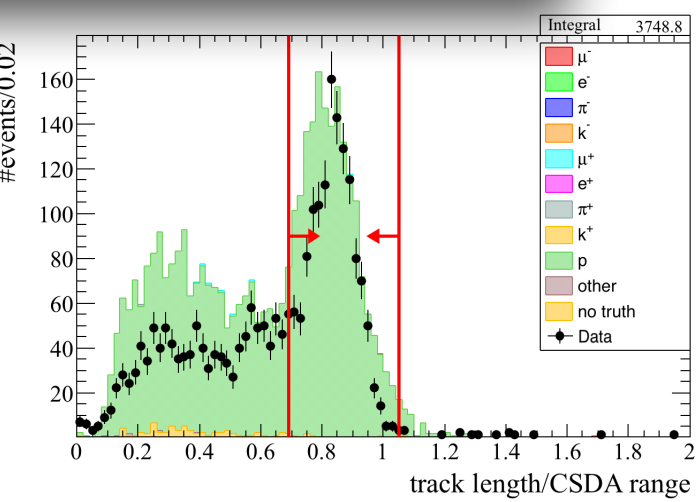


- Reproduce Heng-Ye's 1GeV/c stopping proton analysis
  1. beam TOF compatible with proton
  2. Δx, Δy at z=0
  3. Δθ at z=0
  4. Length/CSDA range (proton)

```
//*****************************************************************
void stoppingProtonSelection::DefineSteps(){
//*****************************************************************

  // Steps must be added in the right order
  // if "true" is added to the constructor of the step,
  // the step sequence is broken if cut is not passed (default is "false")
  AddStep(StepBase::kAction, "find main track",    new FindBeamTrackAction());
  AddStep(StepBase::kCut,    "beam protom",        new BeamProtonCut());
  AddStep(StepBase::kCut,    "beam track in TPC",  new CandidateExistsCut());
  AddStep(StepBase::kCut,    "seltrk angle cut",   new BeamProtonAngleCut());
  AddStep(StepBase::kCut,    "proton CSDA range",  new ProtonCSDARangeCut());

  SetBranchAlias(0,"trunk");
}
```







Example of action

primary information

derived information

```
//************************************************
bool FindBeamTrackAction::Apply(AnaEventC& event, ToyBoxB& boxB) const{
//************************************************

  // Cast the ToyBox to the appropriate type
  ToyBoxPD& box = *static_cast<ToyBoxPD*>(&boxB);

  // Get the array of tracks from the event
  AnaParticleB** tracks = static_cast<AnaEventB*>(&event)->Particles;
  int nTracks           = static_cast<AnaEventB*>(&event)->nParticles;

  // Get the beam information
  AnaBeam* beam = static_cast<AnaBeam*>(static_cast<AnaEventB*>(&event)->Beam);

  // And check that a beam particle exists
  if (!beam->BeamParticle) return true;

  // Loop over reconstructed tracks
  Int_t ncand=0;
  for (Int_t i=0;i<nTracks; ++i){
    AnaParticle* part = static_cast<AnaParticle*>(tracks[i]);

    // In MC just get the beam slice particle (
    if (event.GetIsMC()){
      if (static_cast<AnaParticle*>(tracks[i])->Charge==-8888){
        box.MainTrack = static_cast<AnaParticle*>(tracks[i]);
        ncand++;
        break;
      }
    }
    // In data get the track that better matches the beam-detectors track
    else{
      Float_t dx = part->PositionStart[0]-beam->BeamParticle->PositionEnd[0];
      Float_t dy = part->PositionStart[1]-beam->BeamParticle->PositionEnd[1];
      if (dx>-5 && dx<25 && dy>-10 && dy<10 && part->PositionStart[2]<100 && part->DirectionStart[2]>0.7 ){
        box.MainTrack = part;   // Save the track into the box
        ncand++;
        break;
      }
    }
  }

  return true;
}
```

Example of action

primary information

derived information

```
//************************************************
bool FindBeamTrackAction::Apply(AnaEventC& event, ToyBoxB& boxB) const{
//************************************************

  // Cast the ToyBox to the appropriate type
  ToyBoxPD& box = *static_cast<ToyBoxPD*>(&boxB);

  // Get the array of tracks from the event
  AnaParticleB** tracks = static_cast<AnaEventB*>(&event)->Particles;
  int nTracks           = static_cast<AnaEventB*>(&event)->nParticles;

  // Get the beam information
  AnaBeam* beam = static_cast<AnaBeam*>(static_cast<AnaEventB*>(&event)->Beam);

  // And check that a beam particle exists
  if (!beam->BeamParticle) return true;

  // Loop over reconstructed tracks
  Int_t ncand=0;
  for (Int_t i=0;i<nTracks; ++i){
    AnaParticle* part = static_cast<AnaParticle*>(tracks[i]);

    // In MC just get the beam slice particle (
    if (event.GetIsMC()){
      if (static_cast<AnaParticle*>(tracks[i])->Charge==-8888){
        box.MainTrack = static_cast<AnaParticle*>(tracks[i]);
        ncand++;
        break;
      }
    }
    // In data get the track that better matches the beam-detectors track
    else{
      Float_t dx = part->PositionStart[0]-beam->BeamParticle->PositionEnd[0];
      Float_t dy = part->PositionStart[1]-beam->BeamParticle->PositionEnd[1];
      if (dx>-5 && dx<25 && dy>-10 && dy<10 && part->PositionStart[2]<100 && part->DirectionStart[2]>0.7 ){
        box.MainTrack = part;   // Save the track into the box
        ncand++;
        break;
      }
    }
  }

  return true;
}
```

Example of action

primary information

derived information

```
//************************************************
bool FindBeamTrackAction::Apply(AnaEventC& event, ToyBoxB& boxB) const{
//************************************************

  // Cast the ToyBox to the appropriate type
  ToyBoxPD& box = *static_cast<ToyBoxPD*>(&boxB);

  // Get the array of tracks from the event
  AnaParticleB** tracks = static_cast<AnaEventB*>(&event)->Particles;
  int nTracks           = static_cast<AnaEventB*>(&event)->nParticles;

  // Get the beam information
  AnaBeam* beam = static_cast<AnaBeam*>(static_cast<AnaEventB*>(&event)->Beam);

  // And check that a beam particle exists
  if (!beam->BeamParticle) return true;

  // Loop over reconstructed tracks
  Int_t ncand=0;
  for (Int_t i=0;i<nTracks; ++i){
    AnaParticle* part = static_cast<AnaParticle*>(tracks[i]);

    // In MC just get the beam slice particle (
    if (event.GetIsMC()){
      if (static_cast<AnaParticle*>(tracks[i])->Charge==-8888){
        box.MainTrack = static_cast<AnaParticle*>(tracks[i]);
        ncand++;
        break;
      }
    }
    // In data get the track that better matches the beam-detectors track
    else{
      Float_t dx = part->PositionStart[0]-beam->BeamParticle->PositionEnd[0];
      Float_t dy = part->PositionStart[1]-beam->BeamParticle->PositionEnd[1];
      if (dx>-5 && dx<25 && dy>-10 && dy<10 && part->PositionStart[2]<100 && part->DirectionStart[2]>0.7 ){
        box.MainTrack = part;   // Save the track into the box
        ncand++;
        break;
      }
    }
  }

  return true;
}
```

Example of action

primary information

derived information

```
//************************************************
bool FindBeamTrackAction::Apply(AnaEventC& event, ToyBoxB& boxB) const{
//************************************************

  // Cast the ToyBox to the appropriate type
  ToyBoxPD& box = *static_cast<ToyBoxPD*>(&boxB);

  // Get the array of tracks from the event
  AnaParticleB** tracks = static_cast<AnaEventB*>(&event)->Particles;
  int nTracks           = static_cast<AnaEventB*>(&event)->nParticles;

  // Get the beam information
  AnaBeam* beam = static_cast<AnaBeam*>(static_cast<AnaEventB*>(&event)->Beam);

  // And check that a beam particle exists
  if (!beam->BeamParticle) return true;

  // Loop over reconstructed tracks
  Int_t ncand=0;
  for (Int_t i=0;i<nTracks; ++i){
    AnaParticle* part = static_cast<AnaParticle*>(tracks[i]);

    // In MC just get the beam slice particle (
    if (event.GetIsMC()){
      if (static_cast<AnaParticle*>(tracks[i])->Charge==-8888){
        box.MainTrack = static_cast<AnaParticle*>(tracks[i]);
        ncand++;
        break;
      }
    }
    // In data get the track that better matches the beam-detectors track
    else{
      Float_t dx = part->PositionStart[0]-beam->BeamParticle->PositionEnd[0];
      Float_t dy = part->PositionStart[1]-beam->BeamParticle->PositionEnd[1];
      if (dx>-5 && dx<25 && dy>-10 && dy<10 && part->PositionStart[2]<100 && part->DirectionStart[2]>0.7 ){
        box.MainTrack = part;   // Save the track into the box
        ncand++;
        break;
      }
    }
  }

  return true;
}
```

Example of action

primary information

derived information

```
//************************************************
bool FindBeamTrackAction::Apply(AnaEventC& event, ToyBoxB& boxB) const{
//************************************************

  // Cast the ToyBox to the appropriate type
  ToyBoxPD& box = *static_cast<ToyBoxPD*>(&boxB);

  // Get the array of tracks from the event
  AnaParticleB** tracks = static_cast<AnaEventB*>(&event)->Particles;
  int nTracks           = static_cast<AnaEventB*>(&event)->nParticles;

  // Get the beam information
  AnaBeam* beam = static_cast<AnaBeam*>(static_cast<AnaEventB*>(&event)->Beam);

  // And check that a beam particle exists
  if (!beam->BeamParticle) return true;

  // Loop over reconstructed tracks
  Int_t ncand=0;
  for (Int_t i=0;i<nTracks; ++i){
    AnaParticle* part = static_cast<AnaParticle*>(tracks[i]);

    // In MC just get the beam slice particle (
    if (event.GetIsMC()){
      if (static_cast<AnaParticle*>(tracks[i])->Charge==-8888){
        box.MainTrack = static_cast<AnaParticle*>(tracks[i]);
        ncand++;
        break;
      }
    }
    // In data get the track that better matches the beam-detectors track
    else{
      Float_t dx = part->PositionStart[0]-beam->BeamParticle->PositionEnd[0];
      Float_t dy = part->PositionStart[1]-beam->BeamParticle->PositionEnd[1];
      if (dx>-5 && dx<25 && dy>-10 && dy<10 && part->PositionStart[2]<100 && part->DirectionStart[2]>0.7 ){
        box.MainTrack = part;   // Save the track into the box
        ncand++;
        break;
      }
    }
  }

  return true;
}
```

distance between each track and the beam track

Example of action

primary information

derived information

```
//************************************************
bool FindBeamTrackAction::Apply(AnaEventC& event, ToyBoxB& boxB) const{
//************************************************

  // Cast the ToyBox to the appropriate type
  ToyBoxPD& box = *static_cast<ToyBoxPD*>(&boxB);

  // Get the array of tracks from the event
  AnaParticleB** tracks = static_cast<AnaEventB*>(&event)->Particles;
  int nTracks           = static_cast<AnaEventB*>(&event)->nParticles;

  // Get the beam information
  AnaBeam* beam = static_cast<AnaBeam*>(static_cast<AnaEventB*>(&event)->Beam);

  // And check that a beam particle exists
  if (!beam->BeamParticle) return true;

  // Loop over reconstructed tracks
  Int_t ncand=0;
  for (Int_t i=0;i<nTracks; ++i){
    AnaParticle* part = static_cast<AnaParticle*>(tracks[i]);

    // In MC just get the beam slice particle (
    if (event.GetIsMC()){
      if (static_cast<AnaParticle*>(tracks[i])->Charge==-8888){
        box.MainTrack = static_cast<AnaParticle*>(tracks[i]);
        ncand++;
        break;
      }
    }
    // In data get the track that better matches the beam-detectors track
    else{
      Float_t dx = part->PositionStart[0]-beam->BeamParticle->PositionEnd[0];
      Float_t dy = part->PositionStart[1]-beam->BeamParticle->PositionEnd[1];
      if (dx>-5 && dx<25 && dy>-10 && dy<10 && part->PositionStart[2]<100 && part->DirectionStart[2]>0.7 ){
        box.MainTrack = part;   // Save the track into the box
        ncand++;
        break;
      }
    }
  }

  return true;
}
```

Save this track in the box

Example of cut

- The track in the TPC that matches the beam-detectors track, and that was saved as box.MainTrack, can now be used in a subsequent step, a cut in this case

primary information

derived information

```
//**************************************************
bool ProtonCSDARangeCut::Apply(AnaEventC& event, ToyBoxB& boxB) const{
//**************************************************

  (void)event;

  // Cast the ToyBox to the appropriate type
  ToyBoxPD& box = *static_cast<ToyBoxPD*>(&boxB);
  if (!box.MainTrack) return false;

  //check if it exists a beam particle
  AnaBeam* beam         = static_cast<AnaBeam*>(static_cast<AnaEventB*>(&event)->Beam);
  AnaParticleMomB* beamPart = beam->BeamParticle;
  if(beamPart){
    Float_t mom = beamPart->Momentum;
    //if (beamPart->TrueObject) mom =  static_cast<AnaTrueParticle*>(beamPart->TrueObject)->Momentum;
    Float_t length = static_cast<AnaParticle*>(box.MainTrack)->Length;
    Float_t csdarange = protoDuneSelUtils::ComputeCSDARange(mom*1000, 2212);
    if (csdarange<=0) return false;
    if (length/csdarange>0.69 && length/csdarange<1.05) return true;
  }

  return false;
}
```

Example of cut

- Other cuts used only primary information

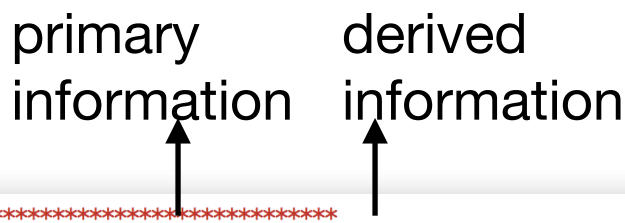
primary information

derived information

```
//**************************************************
bool BeamProtonCut::Apply(AnaEventC& event, ToyBoxB& boxB) const{
//**************************************************

  (void)boxB;

  // Get the beam info
  AnaBeam* beam = static_cast<AnaBeam*>(static_cast<AnaEventB*>(&event)->Beam);

  // In MC require to be a true proton
  if (event.GetIsMC()){
    if (beam->BeamParticle){
      if (beam->BeamParticle->TrueObject)
        if (static_cast<AnaTrueParticle*>(beam->BeamParticle->TrueObject)->PDG==2212) return true;
    }
    return false;
  }
  // In data apply a cut in the TOF
  else{
    if (beam->TOF>170 && beam->TOF<210) return true;
    else return false;
  }
}
```

Example of cut

- Other cuts used only primary information




```
//**************************************************
bool BeamProtonCut::Apply(AnaEventC& event, ToyBoxB& boxB) const{
//**************************************************

  (void)boxB;

  // Get the beam info
  AnaBeam* beam = static_cast<AnaBeam*>(static_cast<AnaEventB*>(&event)->Beam);

  // In MC require to be a true proton
  if (event.GetIsMC()){
    if (beam->BeamParticle){
      if (beam->BeamParticle->TrueObject)
        if (static_cast<AnaTrueParticle*>(beam->BeamParticle->TrueObject)->PDG==2212) return true;
    }
    return false;
  }
  // In data apply a cut in the TOF
  else{
    if (beam->TOF>170 && beam->TOF<210) return true;
    else return false;
  }
}
```

Example of cut

- Other cuts used only primary information

primary information

derived information

```
//**************************************************
bool BeamProtonCut::Apply(AnaEventC& event, ToyBoxB& boxB) const{
//**************************************************

  (void)boxB;

  // Get the beam info
  AnaBeam* beam = static_cast<AnaBeam*>(static_cast<AnaEventB*>(&event)->Beam);

  // In MC require to be a true proton
  if (event.GetIsMC()){
    if (beam->BeamParticle){
      if (beam->BeamParticle->TrueObject)
        if (static_cast<AnaTrueParticle*>(beam->BeamParticle->TrueObject)->PDG==2212) return true;
    }
    return false;
  }
  // In data apply a cut in the TOF
  else{
    if (beam->TOF>170 && beam->TOF<210) return true;
    else return false;
  }
}
```

Example of cut

- Other cuts used only primary information

primary information

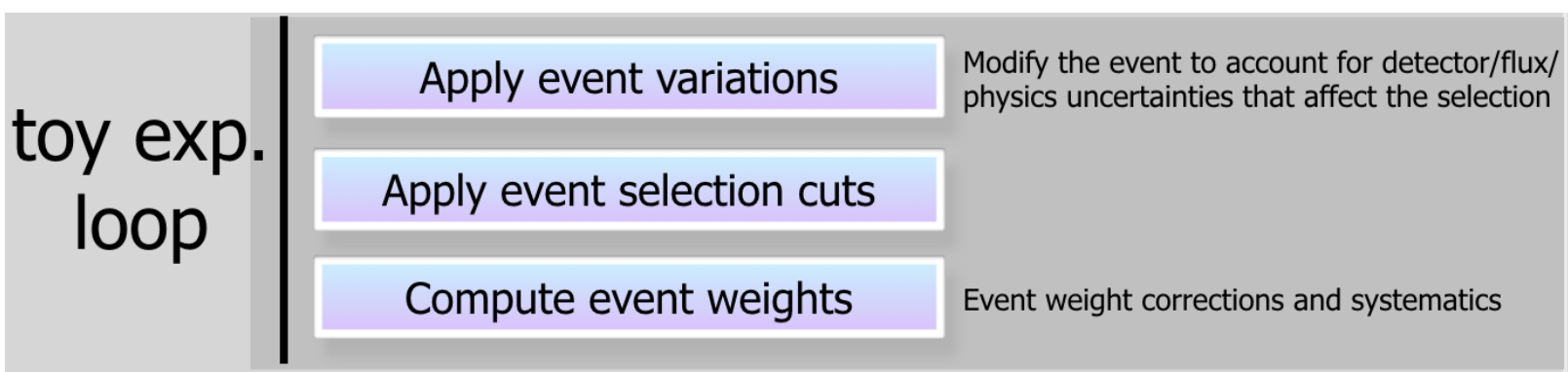
derived information

```
//**************************************************
bool BeamProtonCut::Apply(AnaEventC& event, ToyBoxB& boxB) const{
//**************************************************

  (void)boxB;

  // Get the beam info
  AnaBeam* beam = static_cast<AnaBeam*>(static_cast<AnaEventB*>(&event)->Beam);

  // In MC require to be a true proton
  if (event.GetIsMC()){
    if (beam->BeamParticle){
      if (beam->BeamParticle->TrueObject)
        if (static_cast<AnaTrueParticle*>(beam->BeamParticle->TrueObject)->PDG==2212) return true;
    }
    return false;
  }
  // In data apply a cut in the TOF
  else{
    if (beam->TOF>170 && beam->TOF<210) return true;
    else return false;
  }
}
```

Systematic error propagation

Systematics

- Full systematic propagation functionality is one of the main HighLAND benefits
- Systematic are propagated numerically by multiple throws (toy experiments)
- Two type of systematic propagation methods
  - **Event Variations**: Modify the input data
  - **Event Weights**: Just a global weight for the event



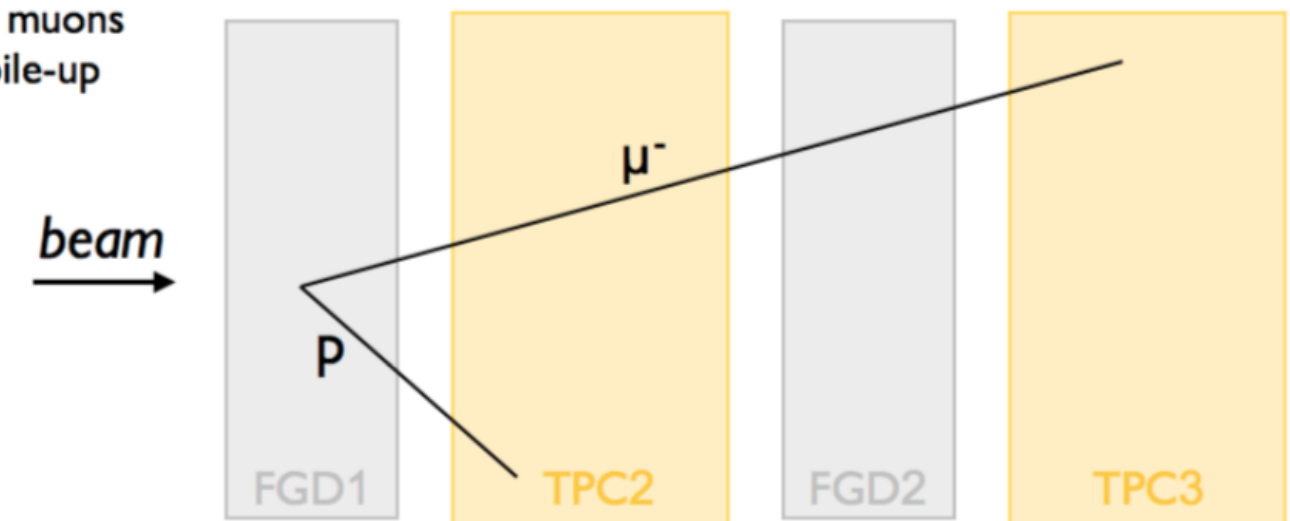
Systematics in T2K

- Example for a selection with a muon and a proton

EXTERNAL BKG:
- out of FGD FV
- sand muons
- cosmic muons
- event pile-up

TPC-FGD:
matching efficiency

μ AND p KINEMATICS:
- charge confusion
- momentum resolution and scale
- B field distortions



FGD1:
- tracking efficiency
- ME efficiency
- PID

TPC2:
- tracking efficiency
- hit efficiency
- PID

MC MODELING:
- pion secondary interactions
- FGD mass

Propagation methods

Some systematic errors in T2K

systematic error source	*propagation model*	*pdf*	*correction*
TPC related			
TPC PID	Reconstructed observable variation	Gaus	yes
TPC cluster efficiency	Efficiency-like	Gaus	no
TPC tracking efficiency	Efficiency-like	Gaus	no
TPC momentum resolution	Reconstructed observable variation	Gaus	yes
TPC charge confusion	Efficiency-like	Gaus	no
B Field distortions	Reconstructed observable variation	Flat	no
TPC momentum scale	Reconstructed observable variation	Gaus	no
FGD1 related			
FGD PID	Reconstructed observable variation	Gaus	yes
FGD tracking efficiency	Efficiency-like	Gaus	no
Michel electron efficiency	Efficiency-like	Gaus	no
FGD-TPC related			
TPC-FGD matching efficiency	Efficiency-like	Gaus	no
Background related			
OOFV background	Normalisation	Gaus	no
Sand muon background	Normalisation	Gaus	no
Pile-up	Normalisation	Gaus	yes
MC modeling related			
Pion secondary interactions	Normalisation	Gaus	no
FGD mass	Normalisation	Gaus	no
beam flux	Normalisation	Gaus	no

Propagation methods

Some systematic errors in T2K

systematic error source	*propagation model*	*pdf*	*correction*	variation
TPC related				
TPC PID	Reconstructed observable variation	Gaus	yes	✓
TPC cluster efficiency	Efficiency-like	Gaus	no	
TPC tracking efficiency	Efficiency-like	Gaus	no	
TPC momentum resolution	Reconstructed observable variation	Gaus	yes	✓
TPC charge confusion	Efficiency-like	Gaus	no	
B Field distortions	Reconstructed observable variation	Flat	no	✓
TPC momentum scale	Reconstructed observable variation	Gaus	no	✓
FGD1 related				
FGD PID	Reconstructed observable variation	Gaus	yes	✓
FGD tracking efficiency	Efficiency-like	Gaus	no	
Michel electron efficiency	Efficiency-like	Gaus	no	
FGD-TPC related				
TPC-FGD matching efficiency	Efficiency-like	Gaus	no	
Background related				
OOFV background	Normalisation	Gaus	no	
Sand muon background	Normalisation	Gaus	no	
Pile-up	Normalisation	Gaus	yes	
MC modeling related				
Pion secondary interactions	Normalisation	Gaus	no	
FGD mass	Normalisation	Gaus	no	
beam flux	Normalisation	Gaus	no	

Propagation methods

Some systematic errors in T2K

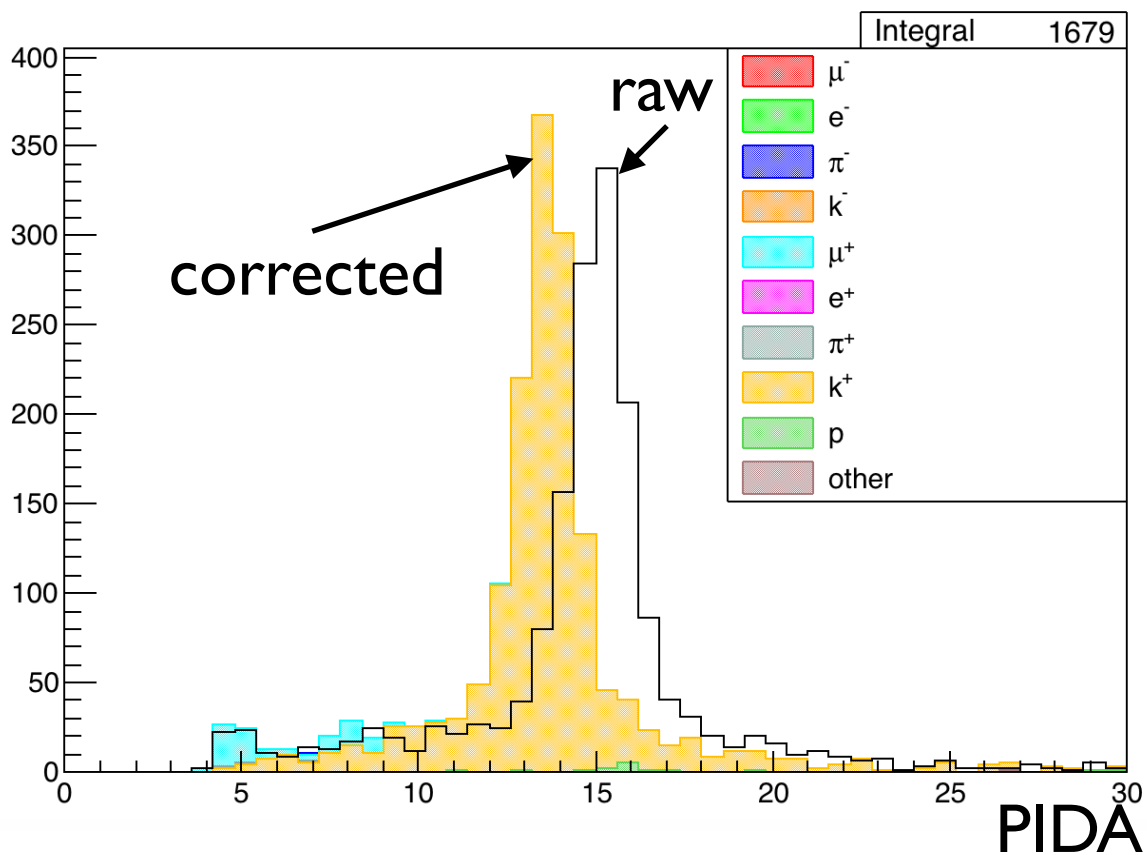
systematic error source	*propagation model*	*pdf*	*correction*	variation	weight
TPC related					
TPC PID	Reconstructed observable variation	Gaus	yes	✓	
TPC cluster efficiency	Efficiency-like	Gaus	no		✓
TPC tracking efficiency	Efficiency-like	Gaus	no		✓
TPC momentum resolution	Reconstructed observable variation	Gaus	yes	✓	
TPC charge confusion	Efficiency-like	Gaus	no		✓
B Field distortions	Reconstructed observable variation	Flat	no	✓	
TPC momentum scale	Reconstructed observable variation	Gaus	no	✓	
FGD1 related					
FGD PID	Reconstructed observable variation	Gaus	yes	✓	
FGD tracking efficiency	Efficiency-like	Gaus	no		✓
Michel electron efficiency	Efficiency-like	Gaus	no		✓
FGD-TPC related					
TPC-FGD matching efficiency	Efficiency-like	Gaus	no		✓
Background related					
OOFV background	Normalisation	Gaus	no		✓
Sand muon background	Normalisation	Gaus	no		✓
Pile-up	Normalisation	Gaus	yes		✓
MC modeling related					
Pion secondary interactions	Normalisation	Gaus	no		✓
FGD mass	Normalisation	Gaus	no		✓
beam flux	Normalisation	Gaus	no		✓

Corrections

- Correct a well known data MC difference to reduce the corresponding systematic
- Example: **dEdxCorrection**
  - Scales the dEdx of each hit by the correction factor and recomputes PIDA

data/dEdx.dat

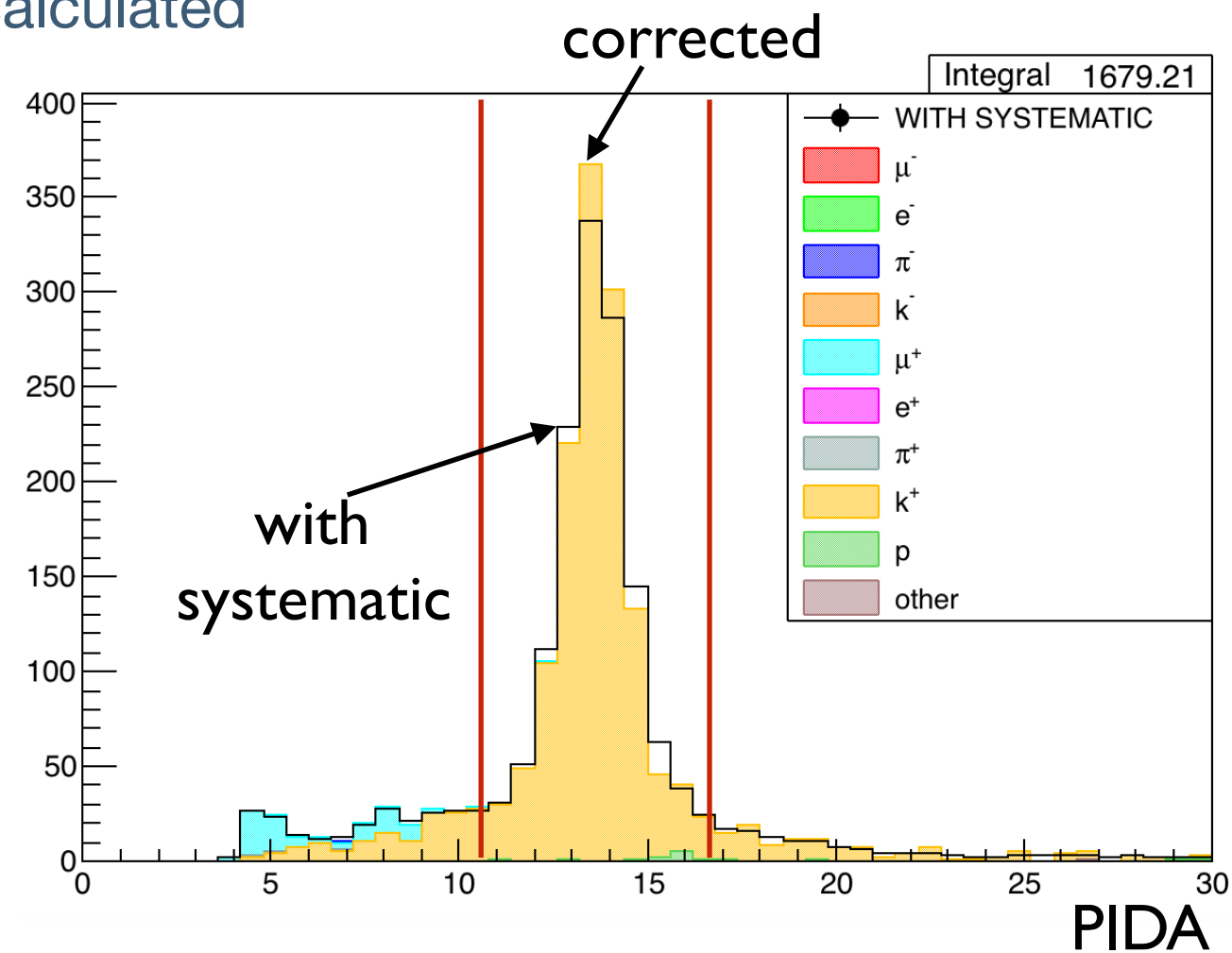
bins of PDG		correction	error on correction
10	12	0.95	0.02
12	14	0.98	0.02
320	322	0.9	0.02
2211	2213	0.8	0.02
210	212	0.9	0.02



dEdxVariation systematic

- The error on the correction is the systematic
- 100 toy experiments. Each toy applies a different correction factor to all hits. Then PIDA is recalculated

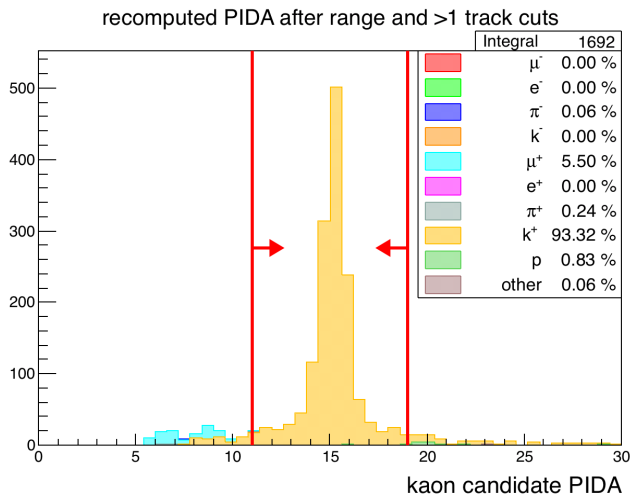
bins of PDG		correction	error on correction
10	12	0.95	0.02
12	14	0.98	0.02
320	322	0.9	0.02
2211	2213	0.8	0.02
210	212	0.9	0.02



The different levels

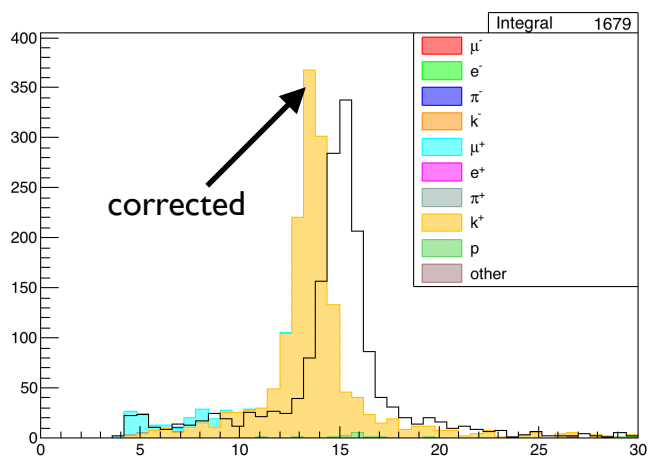
- The three levels are available in the HighLAND output file

raw PIDA



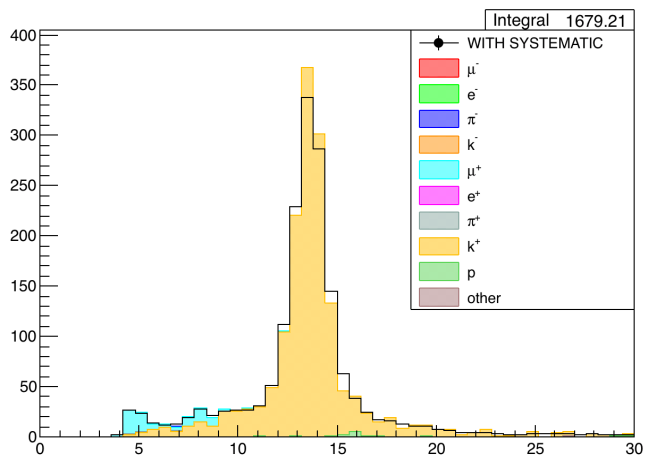
single value for each event

corrected PIDA



single value for each event

systematic propagated PIDA

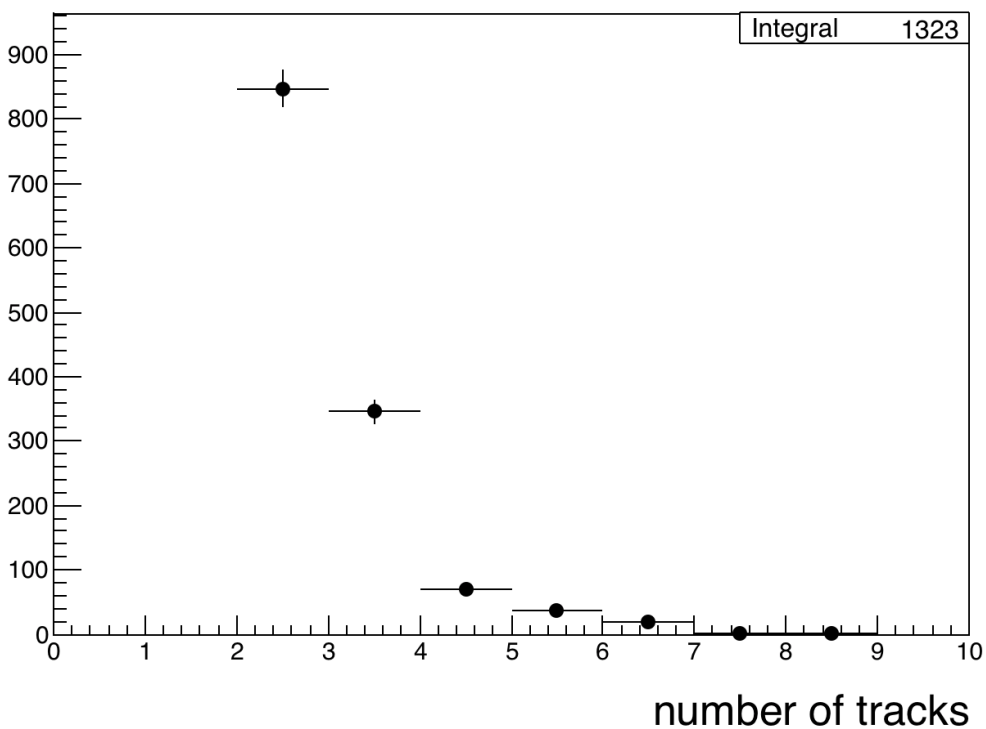


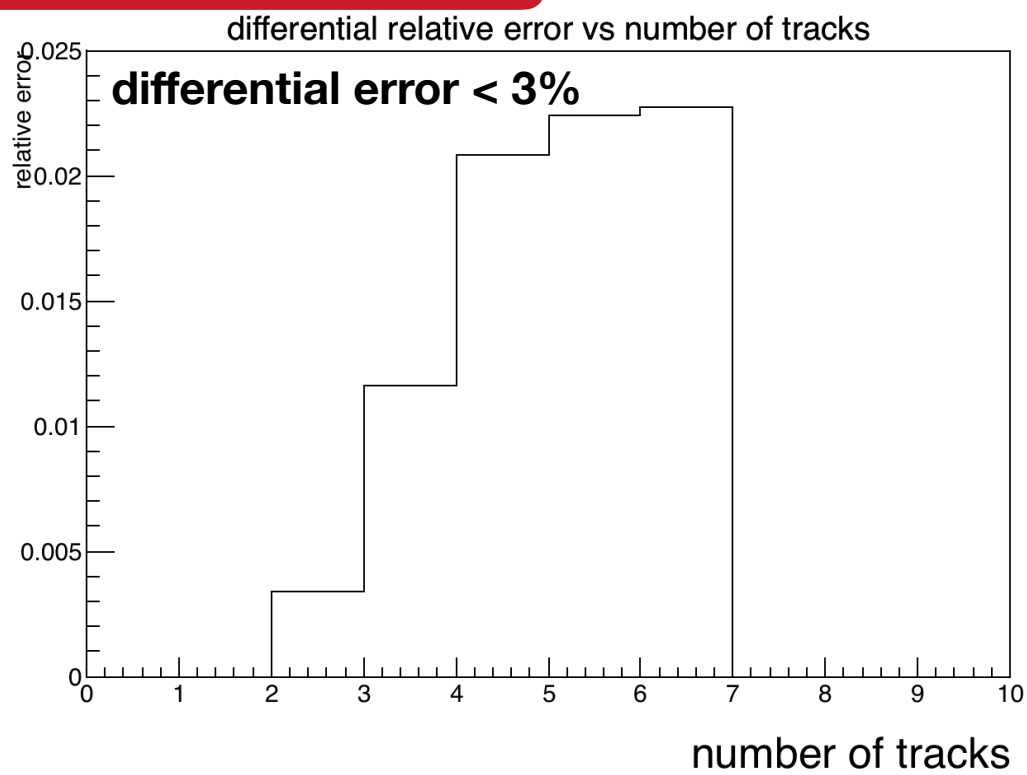
100 values for each event

Effect on the selection

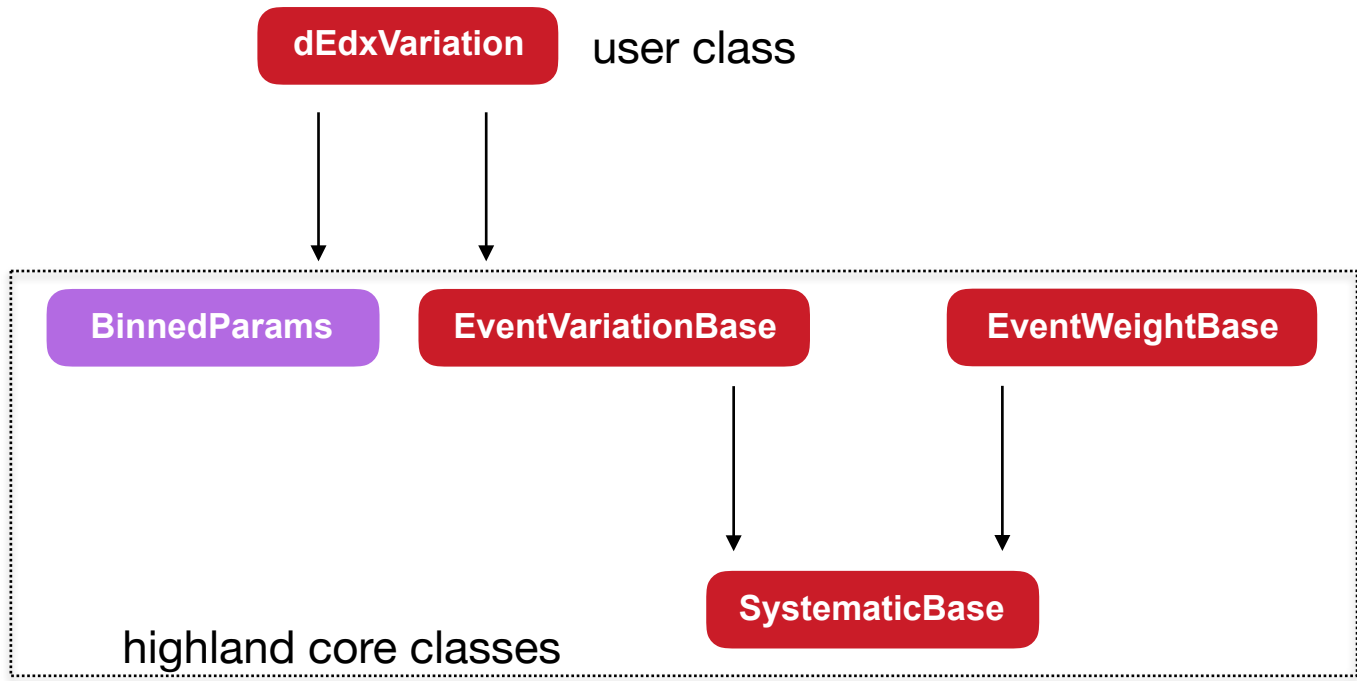
- When the PIDA cut is applied the dEdx systematic has an effect on the number of selected events
  - Integrated: **0.3%**
  - Differential: **< 3%**

for events passing all cuts, including PIDA





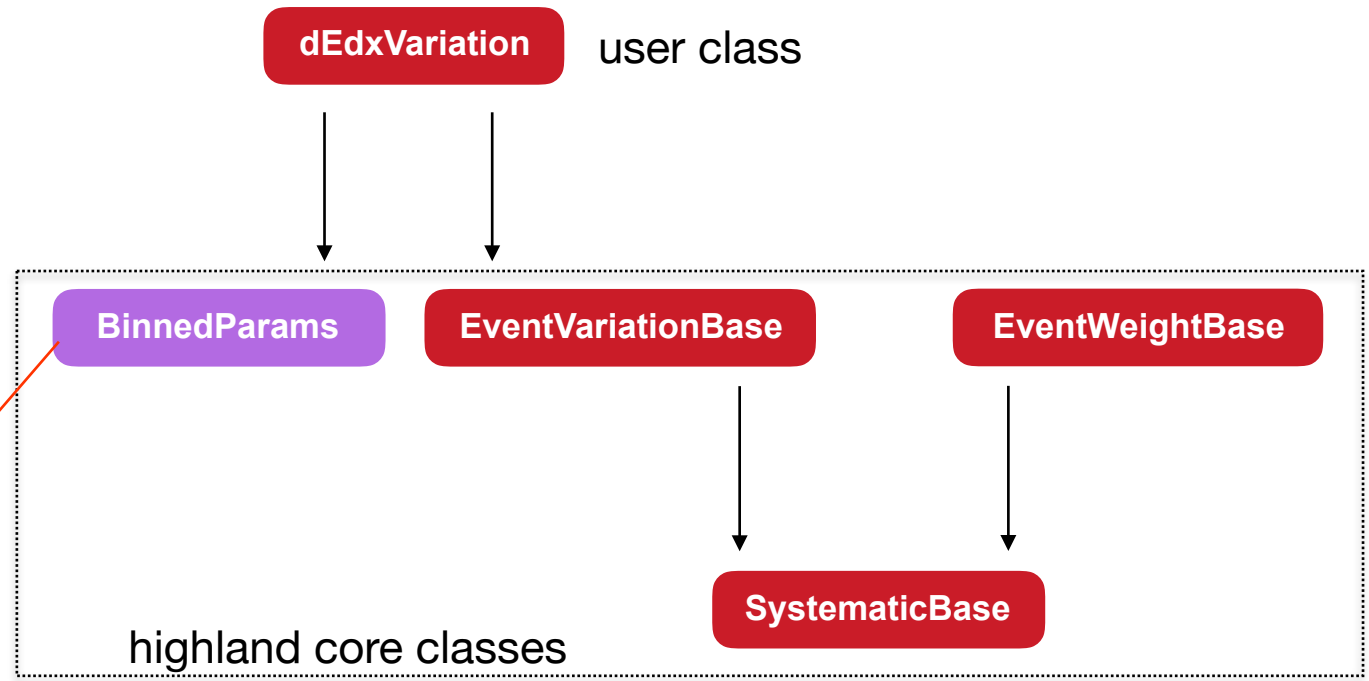
Systematic implementation



SystematicBase

```
/// The name of this systematic.
std::string _name;

/// The index of this systematic (needed by SystematicsManager);
Int_t _index;

/// The type of this systematic (variation, weight or flux)
TypeEnum _type;

/// Is this systematic enabled ?
bool _enabled;

/// The PDF use for the systematic parameter scan
PDFEnum _PDF;

/// the number of systematic parameters
UInt_t _nParameters;
```

```
enum PDFEnum {
  kGaussian = 0,
  kUniform,
  kBinomial,
  kMultinomial,
  kUnknownPDF
}; //!
```



utility class that helps in dealing with data files containing systematic parameters

bins of PDG		correction	error on correction
10	12	0.95	0.02
12	14	0.98	0.02
320	322	0.9	0.02
2211	2213	0.8	0.02
210	212	0.9	0.02

```
enum PDFEnum {
  kGaussian = 0,
  kUniform,
  kBinomial,
  kMultinomial,
  kUnknownPDF
}; //!
```

SystematicBase

```
/// The name of this systematic.
std::string _name;

/// The index of this systematic (needed by SystematicsManager);
Int_t _index;

/// The type of this systematic (variation, weight or flux)
TypeEnum _type;

/// Is this systematic enabled ?
bool _enabled;

/// The PDF use for the systematic parameter scan
PDFEnum _PDF;

/// the number of systematic parameters
UInt_t _nParameters;
```

The header file

mandatory

optional
to gain in speed

```
#ifndef dEdxVariation_h
#define dEdxVariation_h

#include "EventVariationBase.hxx"
#include "BinnedParams.hxx"

/// This systematic smears the CT of each TPC track segment
class dEdxVariation: public EventVariationBase, public BinnedParams{
public:

  /// Instantiate the PID systematic. nbins is the number of
  /// bins in the PDF
  dEdxVariation();

  virtual ~dEdxVariation(){}

  /// Apply the systematic
  virtual void Apply(const ToyExperiment& toy, AnaEventC& event);

  /// Undo  the systematic variations done by ApplyVariation. This is faster tha reseting the full Spill
  bool UndoSystematic(AnaEventC& event);


protected:

  /// Is this track relevant for this systematic ?
  bool IsRelevantRecObject(const AnaEventC& event, const AnaRecObjectC& track) const;

  /// Get the TrackGroup IDs array for this systematic
  Int_t GetRelevantRecObjectGroups(const SelectionBase& sel,     Int_t* IDs) const;

};

#endif
```

Systematic parameters

```
//********************************************************************
dEdxVariation::dEdxVariation(): EventVariationBase(),
                                BinnedParams(std::string(getenv("PROTODUNEEXAMPLEANALYSISROOT"))+"/data",
                                             "dEdx",
                                             BinnedParams::k1D_SYMMETRIC){
//********************************************************************

  // Read the systematic source parameters from the data files
  SetNParameters(GetNBins());
}
```

$PROTODUNEEXAMPLEANALYSISROOT/data/dEdx.dat

PDG		mean_corr	mean_var
10	12	0.95	0.02
12	14	0.98	0.02
320	322	0.9	0.02
2211	2213	0.8	0.02
210	212	0.9	0.02

```
This are the type enums the BinnedParams class can have

For standard weight systematics:
- k1D_SYMMETRIC:       the systematic depends on a single observable (i.e. momentum) and its error is symmetric
- k2D_SYMMETRIC:       the systematic depends on two observables (i.e. momentum and angle) and its error is symmetric
- k3D_SYMMETRIC:       the systematic depends on three observables and its error is symmetric
- k1D_SYMMETRIC_NOMEAN:   the systematic depends on a single observable (i.e. momentum) and its error is symmetric. The correction is not specified
- k2D_SYMMETRIC_NOMEAN:   the systematic depends on two observables (i.e. momentum and angle) and its error is symmetric. The correction is not specified
- k3D_SYMMETRIC_NOMEAN:   the systematic depends on three observables and its error is symmetric. The correction is not specified


For efficiency-like weight systematics:
- k1D_EFF_SYMMETRIC:   the systematic depends on a single observable (i.e. momentum) and its error is symmetric
- k2D_EFF_SYMMETRIC,   the systematic depends on two observables (i.e. momentum and angle) and its error is symmetric
- k3D_EFF_SYMMETRIC,   the systematic depends on three observables and its error is symmetric
- k1D_EFF_ASSYMMETRIC: the systematic depends on a single observable (i.e. momentum) and its error is assymmetric
- k2D_EFF_ASSYMMETRIC: the systematic depends on two observables (i.e. momentum and angle) and its error is assymmetric
- k3D_EFF_ASSYMMETRIC, the systematic depends on three observables and its error is assymmetric
```

Relevant objects

- This method is called only once per event at initialisation level
- Only relevant objects for this systematics are saved into the SystBox
- In this way, for each toy experiment, the loop over objects can be much faster

```
//********************************************************
bool dEdxVariation::IsRelevantRecObject(const AnaEventC& event, const AnaRecObjectC& track) const{
  //********************************************************

  (void)event;

  // True track should always exist
  if (!track.TrueObject) return false;

  AnaTrueParticleB* truePart = static_cast<AnaTrueParticleB*>(track.TrueObject);

  // only consider true protons, pions, muons and electrons
  if      (abs(truePart->PDG) == 211 ) return true;
  else if (abs(truePart->PDG) == 2212) return true;
  else if (abs(truePart->PDG) == 13)   return true;
  else if (abs(truePart->PDG) == 11)   return true;
  else if (abs(truePart->PDG) == 321)  return true;

  return false;
}
```

The systematic propagation

```
//*************************************************************************
void dEdxVariation::Apply(const ToyExperiment& toy, AnaEventC& event){
//*************************************************************************

  // Get the SystBox for this event
  SystBoxB* box = GetSystBox(event);

  // Loop over all relevant tracks for this variation
  for (Int_t itrk = 0; itrk < box->nRelevantRecObjects; itrk++){

    // Get the AnaPArticle
    AnaParticle* part = static_cast<AnaParticle*>(box->RelevantRecObjects[itrk]);

    // The un-corrected particle
    const AnaParticle* original = static_cast<const AnaParticle*>(part->Original);

    if (!part->TrueObject)            continue; //?
    if (!original)                    continue; //?

    // Get the true particle associated to the reconstructed particle
    AnaTrueParticleB* truePart = static_cast<AnaTrueParticleB*>(part->TrueObject);

    Float_t mean_corr;   // the correction
    Float_t mean_var;    // the error on the correction (this is the systematic)
    Int_t mean_index;    // the index of the bin in the data file with the systematic parameters

    // Get the systematic parameters for this particle type (PDG)
    if (!GetBinValues(abs(truePart->PDG), mean_corr,  mean_var,  mean_index))  return;


    for (Int_t i=0;i<3;i++){       // Loop over wire planes
      for (Int_t j=0;j<part->NHitsPerPlane[i];j++){       // Loop over hits in that plane
        part->dEdx[i][j] = original->dEdx[i][j] *(1 +  toy.GetToyVariations(_index)->Variations[mean_index]*mean_var/mean_corr);
      }
    }
  }
}
```

PDG		mean_corr	mean_var
10	12	0.95	0.02
12	14	0.98	0.02
320	322	0.9	0.02
2211	2213	0.8	0.02
210	212	0.9	0.02

$$x' = x \cdot \left(1 + \delta \cdot \frac{\sigma}{\mu}\right)$$

Undo the variation

- The next toy experiment should do variations from the nominal values, not from the previous toy. Two options:
  - Reset the entire event, clone again the original event **slow**
  - Just reset the observables the systematic has changed **fast**

```
//********************************************************************
bool dEdxVariation::UndoSystematic(AnaEventC& event){
//********************************************************************

  // Get the SystBox for this event
  SystBoxB* box = GetSystBox(event);

  // Loop over relevant objects for this systematics
  for (Int_t itrk=0;itrk<box->nRelevantRecObjects;itrk++){

    // Get the particle and its original
    AnaParticle* part = static_cast<AnaParticle*>(box->RelevantRecObjects[itrk]);
    const AnaParticle* original = static_cast<const AnaParticle*>(part->Original);
    if (!original)   continue;

    // Revert the dEdx values to the original ones
    for (Int_t i=0;i<3;i++){
      for (Int_t j=0;j<part->NHitsPerPlane[i];j++){
        part->dEdx[i][j] = original->dEdx[i][j];
      }
    }

  }

  // Don't reset the spill to corrected
  return false;
}
```

The systematic propagation

```
//*************************************************************************
void dEdxVariation::Apply(const ToyExperiment& toy, AnaEventC& event){
//*************************************************************************

  // Get the SystBox for this event
  SystBoxB* box = GetSystBox(event);

  // Loop over all relevant tracks for this variation
  for (Int_t itrk = 0; itrk < box->nRelevantRecObjects; itrk++){

    // Get the AnaPArticle
    AnaParticle* part = static_cast<AnaParticle*>(box->RelevantRecObjects[itrk]);

    // The un-corrected particle
    const AnaParticle* original = static_cast<const AnaParticle*>(part->Original);

    if (!part->TrueObject)            continue; //?
    if (!original)                    continue; //?

    // Get the true particle associated to the reconstructed particle
    AnaTrueParticleB* truePart = static_cast<AnaTrueParticleB*>(part->TrueObject);

    Float_t mean_corr;   // the correction
    Float_t mean_var;    // the error on the correction (this is the systematic)
    Int_t mean_index;    // the index of the bin in the data file with the systematic parameters

    // Get the systematic parameters for this particle type (PDG)
    if (!GetBinValues(abs(truePart->PDG), mean_corr,  mean_var,  mean_index))  return;


    for (Int_t i=0;i<3;i++){       // Loop over wire planes
      for (Int_t j=0;j<part->NHitsPerPlane[i];j++){       // Loop over hits in that plane
        part->dEdx[i][j] = original->dEdx[i][j] *(1 +  toy.GetToyVariations(_index)->Variations[mean_index]*mean_var/mean_corr);
      }
    }
  }
}
```

PDG		mean_corr	mean_var
10	12	0.95	0.02
12	14	0.98	0.02
320	322	0.9	0.02
2211	2213	0.8	0.02
210	212	0.9	0.02

$$x' = x \cdot \left(1 + \delta \cdot \frac{\sigma}{\mu}\right)$$

The systematic propagation

```
//*************************************************************************
void dEdxVariation::Apply(const ToyExperiment& toy, AnaEventC& event){
//*************************************************************************

  // Get the SystBox for this event
  SystBoxB* box = GetSystBox(event);

  // Loop over all relevant tracks for this variation
  for (Int_t itrk = 0; itrk < box->nRelevantRecObjects; itrk++){

    // Get the AnaPArticle
    AnaParticle* part = static_cast<AnaParticle*>(box->RelevantRecObjects[itrk]);

    // The un-corrected particle
    const AnaParticle* original = static_cast<const AnaParticle*>(part->Original);

    if (!part->TrueObject)            continue; //?
    if (!original)                    continue; //?

    // Get the true particle associated to the reconstructed particle
    AnaTrueParticleB* truePart = static_cast<AnaTrueParticleB*>(part->TrueObject);

    Float_t mean_corr;   // the correction
    Float_t mean_var;    // the error on the correction (this is the systematic)
    Int_t mean_index;    // the index of the bin in the data file with the systematic parameters

    // Get the systematic parameters for this particle type (PDG)
    if (!GetBinValues(abs(truePart->PDG), mean_corr,  mean_var,  mean_index))  return;


    for (Int_t i=0;i<3;i++){       // Loop over wire planes
      for (Int_t j=0;j<part->NHitsPerPlane[i];j++){       // Loop over hits in that plane
        part->dEdx[i][j] = original->dEdx[i][j] *(1 +  toy.GetToyVariations(_index)->Variations[mean_index]*mean_var/mean_corr);
      }
    }
  }
}
```

PDG		mean_corr	mean_var
10	12	0.95	0.02
12	14	0.98	0.02
320	322	0.9	0.02
2211	2213	0.8	0.02
210	212	0.9	0.02

$$x' = x \cdot \left(1 + \delta \cdot \frac{\sigma}{\mu}\right)$$

systematic index

parameter index for this systematic

Filling toy experiments

already implemented

- For each toy experiment the ToyExperiment class is filled:
  - A random throw for each parameter in each systematic

This method fills the δ
for each systematic parameter

$$x' = x \cdot \left(1 + \delta \cdot \frac{\sigma}{\mu}\right)$$

```
//*************************************************************
void baseToyMaker::FillToyExperiment(ToyExperiment& toy){
//*************************************************************

  // Set the same weight (1) for al toys. This will be later normalized to the number of toys
  Float_t weight = 1.;

  // Loop over all systematics
  for (UInt_t isyst = 0; isyst<NMAXSYSTEMATICS;isyst++){
    SystematicBase* syst = _systematics[isyst];
    if (!syst) continue;

    // Create the proper structure for the ToyExperiment adding each of the ToyVariations
    toy.AddToyVariation(syst->GetIndex(), syst->GetNParameters());

    // Loop over parameters for this systematic
    for (UInt_t ipar = 0;ipar<syst->GetNParameters();ipar++){
      Float_t var = 0;
      // When the option _zero_var is enabled all variations are 0
      if (!_zero_var){
        if (_individualRandomGenerator){
          if      (syst->PDF() == SystematicBase::kUniform)  var = _RandomGenerators[isyst].Uniform(0.,1.);
          else if (syst->PDF() == SystematicBase::kGaussian) var = _RandomGenerators[isyst].Gaus(0.,1.);
        }
        else{
          if      (syst->PDF() == SystematicBase::kUniform)  var = _RandomGenerator.Uniform(0.,1.);
          else if (syst->PDF() == SystematicBase::kGaussian) var = _RandomGenerator.Gaus(0.,1.);
        }
      }
      toy.SetToyVariation(isyst,ipar,var,weight);
    }
  }
}
```

systematic index

δ

parameter index
for that systematic

weight of this toy (in general the same for all toys)

What the user should implement ?

- For the systematic, only header, constructor and Apply method

```
#ifndef dEdxVariation_h
#define dEdxVariation_h

#include "EventVariationBase.hxx"
#include "BinnedParams.hxx"

/// This systematic smears the CT of each TPC track segment
class dEdxVariation: public EventVariationBase, public BinnedParams{
public:

  /// Instantiate the PID systematic. nbins is the number of
  /// bins in the PDF
  dEdxVariation();

  virtual ~dEdxVariation(){}

  /// Apply the systematic
  virtual void Apply(const ToyExperiment& toy, AnaEventC& event);

  /// Undo  the systematic variations done by ApplyVariation. This is faster tha reseting the full Spill
  bool UndoSystematic(AnaEventC& event);


protected:

  /// Is this track relevant for this systematic ?
  bool IsRelevantRecObject(const AnaEventC& event, const AnaRecObjectC& track) const;

  /// Get the TrackGroup IDs array for this systematic
  Int_t GetRelevantRecObjectGroups(const SelectionBase& sel,     Int_t* IDs) const;

};

#endif
```

```
//********************************************************************
dEdxVariation::dEdxVariation(): EventVariationBase(),
                                BinnedParams(std::string(getenv("PROTODUNEEXAMPLEANALYSISROOT"))+"/data",
                                             "dEdx",
                                             BinnedParams::k1D_SYMMETRIC){
//********************************************************************

  // Read the systematic source parameters from the data files
  SetNParameters(GetNBins());
}
```

```
//********************************************************************
void dEdxVariation::Apply(const ToyExperiment& toy, AnaEventC& event){
//********************************************************************

  // Get the SystBox for this event
  SystBoxB* box = GetSystBox(event);

  // Loop over all relevant tracks for this variation
  for (Int_t itrk = 0; itrk < box->nRelevantRecObjects; itrk++){

    // Get the AnaPArticle
    AnaParticle* part = static_cast<AnaParticle*>(box->RelevantRecObjects[itrk]);

    // The un-corrected particle
    const AnaParticle* original = static_cast<const AnaParticle*>(part->Original);

    if (!part->TrueObject)            continue; //?
    if (!original)                    continue; //?

    // Get the true particle associated to the reconstructed particle
    AnaTrueParticleB* truePart = static_cast<AnaTrueParticleB*>(part->TrueObject);

    Float_t mean_corr;   // the correction
    Float_t mean_var;    // the error on the correction (this is the systematic)
    Int_t mean_index;    // the index of the bin in the data file with the systematic parameters

    // Get the systematic parameters for this particle type (PDG)
    if (!GetBinValues(abs(truePart->PDG), mean_corr,  mean_var,  mean_index))  return;


    for (Int_t i=0;i<3;i++){       // Loop over wire planes
      for (Int_t j=0;j<part->NHitsPerPlane[i];j++){       // Loop over hits in that plane
        part->dEdx[i][j] = original->dEdx[i][j] *(1 +  toy.GetToyVariations(_index)->Variations[mean_index]*mean_var/mean_corr);
      }
    }
  }
}
```

- The rest is optional or already provided with the framework

- Obviously the user should implement his analysis algorithm and tell the system with selections and systematics to use

```
class protoDuneExampleAnalysis: public baseAnalysis {
 public:
  protoDuneExampleAnalysis(AnalysisAlgorithm* ana=NULL);
  virtual ~protoDuneExampleAnalysis(){}

  //---- These are mandatory functions
  void DefineSelections();
  void DefineCorrections();
  void DefineSystematics();
  void DefineConfigurations();
  void DefineMicroTrees(bool addBase=true);
  void DefineTruthTree();

  void FillMicroTrees(bool addBase=true);
  void FillToyVarsInMicroTrees(bool addBase=true);

  bool CheckFillTruthTree(const AnaTrueVertex& vtx);

  using baseAnalysis::FillTruthTree;
  void FillTruthTree(const AnaTrueVertex& vtx);
  //--------------------
```

- Add the systematic to the EventVariationManager

```
//********************************************************************
void protoDuneExampleAnalysis::DefineSystematics(){
//********************************************************************

  // Some systematic are defined in baseAnalysis
  baseAnalysis::DefineSystematics();

  evar().AddEventVariation(SystId::kdEdx,          "dEdx",          new dEdxVariation());
}
```

- Add the selection to the SelectionManager

```
//********************************************************************
void protoDuneExampleAnalysis::DefineSelections(){
//********************************************************************

  /* In this method the user will add to the SelectionManager (accessed by  sel() ) the selections to be run,
     defined in other files. In general an analysis has a single selection, which could have multiple branches.
     Each of the branches is in fact a different selection (different cuts and actions), but defining them as branches
     we ussualy gain in speed and simplicity since the steps that are common are applied only once.
     Sometimes the user does not want to expend time merging two selections in a single one (as branches), but preffers to run
     both independently. This is in general done for quick checks (are the selections mutually exclusive ?, etc).
     This is possible in highland2. An example on how to do that is shown below.
  */

  // Add selections to the SelectionManager provided:
  // - Name of the selection
  // - Title, the one dumped by the DrawingTools::DumpSelections() method. It is an explaination of the selection
  // - Pointer to the selection. The argument in the constructor (false) indicates the
  //   step sequence is not broken when a cut is not passed. In this way we can save intermediate information for events
  //   not passing the entire selection

  sel().AddSelection("stoppingProtonSelection",  "protoDuneExample selection",     new stoppingProtonSelection(false));     // true/false for forcing break
  sel().AddSelection("stoppingKaonSelection",    "protoDuneExample selection",     new stoppingKaonSelection(false));     // true/false for forcing break
  sel().AddSelection("stoppingMuonSelection",    "protoDuneExample selection",     new stoppingMuonSelection(false));
}
```

The path forward

- Code preparation
  - Migration to cmake **almost done**
  - Upgrade LArSoftConverter to ProtoDUNE production 2 **almost done**
- Git repositories
  - Give access to git repository in Valencia to few experts
  - new repository for user code in FNAL
- Implementation of ProtoDUNE selections and systematics
  - Get in contact with analyzers who want to start propagating systematics and understand their needs
  - Migrate their selections to HighLAND
  - Proto implementation of some systematics

- Thoughts from the analysis group

https://docs.google.com/document/d/15akjuTH7qI4mG-0rDTIsMlZUq1SazquFx-TAG25qvTw/edit

Beam composition
Beam momentum
Beam-TPC Energy loss
Signal model
Vertex Identification/Position
space point -> reconstructed object not done correctly
Signal definition
muon contamination from pion decay
Cosmic background
Incorrect classification of track-like particles as showers
Fiducial Volume definition
Stitching across APAs
EField value
APAs Alignment
Wire response (cross-talk in wires?)
Gain of the channel
ADC to charge conversion
signal/noise
T0 estimation → synchronization with beam clock?
Lifetime ? f(T, time, y)
Drift velocity
Recombination / dEdx : f(E, t, theta)

In T2K

systematic error source
TPC related
TPC PID
TPC cluster efficiency
TPC tracking efficiency
TPC momentum resolution
TPC charge confusion
B Field distortions
TPC momentum scale
FGD1 related
FGD PID
FGD tracking efficiency
Michel electron efficiency
FGD-TPC related
TPC-FGD matching efficiency
Background related
OOFV background
Sand muon background
Pile-up
MC modeling related
Pion secondary interactions
FGD mass
beam flux

backup

- Beam Composition: weights
- Beam Momentum
    - WEIGHTABLE? Build a prediction in the particle gun with a spread of energies consistent with whatever we predict from a beamline simulation which is dedicated and separate. Then we can slosh events around.
    - JC: The particle gun already has a spread mentioned.
    - JC: the CERN group has a dedicated separate simulation. There was an idea to simulate the whole thing. => Haven't heard-- we could check in on this? Clarify what if any plan was here.
- Energy loss between beam and TPC

- Signal model: Misidentifying or missing pions/protons → Selection Uncertainty
  - Absorption, Charge exchange model uncertainties:
  - Geant4 predicted hadronic final state composition (number of protons, neutrons,
  - Final state particle kinematics (outgoing momentum, angle to each particle)
  - => Where do we think the signal model is potentially insufficient?
  - Mine DUET paper
  - Ron Ransome's talk:
  - Then, develop a suitable technique with weighting or particle gun stitching.
  - Background model? Elastic scattering.
  - => Same as above.

- Reconstruction error: Vertex Identification/Position
- This will couple to both the selection uncertainty and the energy estimations in a few ways:
- Stop ‘too short’
- Stop ‘too late’ (i.e. along daughter particle)
- Miss elastic scattering -- Need to understand energy lost to nucleus

Systematics

- Systematics are propagated numerically using toy- experiments (pseudo-experiments or virtual analyses)
- Each toy-experiment is defined by a set of random throws (one for each systematic parameter)
- The covariance of the number of events selected in a given bin is computed in the usual way:

$$C_{ij} = \frac{1}{N_{toys}} \sum_{t=1}^{N_{toys}} (N_i^t - \overline{N}_i)(N_j^t - \overline{N}_j)$$

events in bin i for toy t

$$N_i^t = \sum_{e=1}^{N_{events}} W_{e,i}^t$$

average over toys

$$\overline{N}_i = \frac{1}{N_{toys}} \sum_{t=1}^{N_{toys}} N_i^t$$

Analysis group concerns

Analysis group's concerns

- In the next slides I address the concerns of the analysis group:
  - new framework to learn
  - need hit level info
  - highland uses CMT and not cmake
  - code sharing and git repository

Concern 1

- **Analyzers are familiar with protoduneana, not with HighLAND, and they will expend a lot of time trying to understand the new framework**

- It takes 5 minutes to download and compile HighLAND
- It is a very light framework, with very few concepts to understand. It will take a day for a student to start producing results
- Migrating an existing analysis to HighLAND will take few days. After that, progress will be much faster
- We can help with the migration !!!

Concern 2

- **All the analyses right now are still working at the hit level (vertex mis-reconstruction studies, elastic scattering tagging, machine learning PID and shower reconstruction, etc.)**
  - HighLAND can run on LArSoft reco files
- **This is highly complicated for protodune, an analyser has first to work with the reco files, then move to highland (or another analysis package) for selection and systematics and then move to a fit/unfolding package for a measurement**
  - Since HighLAND works with any input format (provided the appropriate converter) moving to higher analysis levels and reduced inputs is completely transparent
  - Ideally fitting/unfolding package should be such that it can accept any input (HighLAND or other)

Concern 3

- **HighLAND uses CMT while protoduneana uses cmake**

- Being HighLAND a separate set of packages it does not matter
- Anyway, we will migrate to cmake next week. It is straight forward. In fact it was already in the to do list

Concern 4

- **Analysers working on similar analyses, how they will commit and share code since the code lives on a different repository?**
- This is how we get existing highland packages

```
git clone https://next.ific.uv.es:8888/HighLAND/psychePolicy.git                 psyche/psychePolicy/v0r0
git clone https://next.ific.uv.es:8888/HighLAND/psycheCore.git                   psyche/psycheCore/v0r0
git clone https://next.ific.uv.es:8888/HighLAND/psycheUtils.git                  psyche/psycheUtils/v0r0

git clone https://next.ific.uv.es:8888/duneHighLAND/psycheDUNEUtils.git          psyche/psycheDUNEUtils/v0r0
git clone https://next.ific.uv.es:8888/duneHighLAND/psycheEventModel.git         psyche/psycheEventModel/v0r0
git clone https://next.ific.uv.es:8888/duneHighLAND/psycheIO.git                 psyche/psycheIO/v0r0
git clone https://next.ific.uv.es:8888/duneHighLAND/psycheSelections.git         psyche/psycheSelections/v0r0

git clone https://next.ific.uv.es:8888/HighLAND/highlandCore.git                 highland2/highlandCore/v0r0
git clone https://next.ific.uv.es:8888/HighLAND/highlandTools.git                highland2/highlandTools/v0r0
git clone https://next.ific.uv.es:8888/HighLAND/highlandDoc.git                  highland2/highlandDoc/v0r0

git clone https://next.ific.uv.es:8888/duneHighLAND/highlandEventModel.git       highland2/highlandEventModel/v0r0
git clone https://next.ific.uv.es:8888/duneHighLAND/highlandUtils.git            highland2/highlandUtils/v0r0
git clone https://next.ific.uv.es:8888/duneHighLAND/highlandCorrections.git      highland2/highlandCorrections/v0r0
git clone https://next.ific.uv.es:8888/duneHighLAND/LArSoftReader.git            highland2/LArSoftReader/v0r0
git clone https://next.ific.uv.es:8888/duneHighLAND/highlandIO.git               highland2/highlandIO/v0r0
git clone https://next.ific.uv.es:8888/duneHighLAND/baseAnalysis.git             highland2/baseAnalysis/v0r0
git clone https://next.ific.uv.es:8888/duneHighLAND/duneExampleAnalysis.git      highland2/duneExampleAnalysis/v0r0
git clone https://next.ific.uv.es:8888/duneHighLAND/protoDuneExampleAnalysis.git highland2/protoDuneExampleAnalysis/v0r0
```

- We can get protoDUNE specific highland packages from the DUNE repository

git clone http://cdcvs.fnal.gov/projects/protoduneHighland highland2/myAnalysis/v0r0

Data reduction

Data Reduction functionality

- **LArSoft files are way too big**
  - I don't have much experience with them but getting them and running on them is not easy
  - Most information in those files is not needed for the analysis
- **The analysis should proceed on files that are manageable**
  - Small size
  - Fast to run over
  - Easy to understand for non-experts in simulation/reconstruction
- HighLAND can read LArSoft files and produce two other file types, much smaller and simpler
- Those files can be used as input for HighLAND analyses

Data Reduction functionality

- The event model can be dumped into a root file in two ways:
  - **MiniTree:** The class AnaEvent is saved
    - **Pros**: Does not need maintenance, changes en the event model are automatically propagated
    - **Cons:** Not very easy to navigate
  - **FlatTree:** Similar to the DUNE AnaTrees**.** The user decides the objects that are saved in the tree and the name of the variables. The output is a flat tree with basic type variables: double, float, int, char, and vectors of them.
    - **Pros**: Very easy to navigate
    - **Cons:** Difficult to maintain. Need to propagate changes in event model
- Both have similar size and running speed

Running directly on LArSoft files

- A list of LArSoft files in a text file: file.list

~50 GB for 230 events

```
/pnfs/dune/tape_backed/dunepro/mcc10/mc/full-reconstructed/02/05/18/93/mcc10_protodune_beam_p3GeV_cosmics_3ms_sce_2_20171229T053321_merged0.root
/pnfs/dune/tape_backed/dunepro/mcc10/mc/full-reconstructed/02/05/18/94/mcc10_protodune_beam_p3GeV_cosmics_3ms_sce_1_20171229T054519_merged0.root
/pnfs/dune/tape_backed/dunepro/mcc10/mc/full-reconstructed/02/05/18/95/mcc10_protodune_beam_p3GeV_cosmics_3ms_sce_8_20171229T054930_merged0.root
/pnfs/dune/tape_backed/dunepro/mcc10/mc/full-reconstructed/02/05/18/96/mcc10_protodune_beam_p3GeV_cosmics_3ms_sce_21_20171229T055045_merged0.root
/pnfs/dune/tape_backed/dunepro/mcc10/mc/full-reconstructed/02/05/18/97/mcc10_protodune_beam_p3GeV_cosmics_3ms_sce_66_20171229T054500_merged0.root
/pnfs/dune/tape_backed/dunepro/mcc10/mc/full-reconstructed/02/05/18/98/mcc10_protodune_beam_p3GeV_cosmics_3ms_sce_24_20171229T061658_merged0.root
```

- Run the example example over that list in dunegpvm03.fnal.gov

```
../Linux-x86_64/RunProtoDuneExampleAnalysis.exe -v -o test_230evt.root  file.list
```

- This is the final output on the screen

```
entry: 230 of 230 (100%) --> 230
time profile --------------
Ini Spill:         2321.84
Ini Bunch:         3.44837
Ini Conf:          0.577693
Ini Toy (v syst):  0.00119042
Ini Sel:           0.00724292
Selection:         0.0511615
End Sel (w syst):  8.36849e-05
End Toy:           0.0272403
End Conf:          4.85944
End Bunch:         0.183686
End Spill:         0.254492
Total:             2331.58
Total wo t:        2331.25
```

most time expended in reading the file

40 minutes for 230 events

MiniTree

- First create a MiniTree from the same list

```
../Linux-x86_64/RunCreateMiniTree.exe -v  -o mini_230evt.root ../../../protoDuneExampleAnalysis/v0r0/cmt/file.list

--------- time profile --------------
230 entries processed in 2291.63 seconds
```

takes 40 minutes

- The MiniTree is much smaller than the initial LArSoft files and can be easily transferred to a laptop

~20 MB (a factor 5000)

most space taken by truth info.
(More than 500 particles/event)

- Now running the analysis is much faster

```
../Linux-x86_64/RunProtoDuneExampleAnalysis.exe -v -o test_mini_230evt.root  ../../../highlandIO/v0r0/cmt/mini_230evt.root
```

```
entry: 230 of 230 (100%) --> 230
time profile --------------
Ini Spill:         2.01702
Ini Bunch:         0.0323527
Ini Conf:          0.00446773
Ini Toy (v syst): 0.000124216
Ini Sel:           7.7486e-05
Selection:         0.0019443
End Sel (w syst): 4.50611e-05
End Toy:           0.00111413
End Conf:          0.0445073
End Bunch:         0.0177789
End Spill:         0.0364242
Total:             2.1564
Total wo t:        2.15586
```

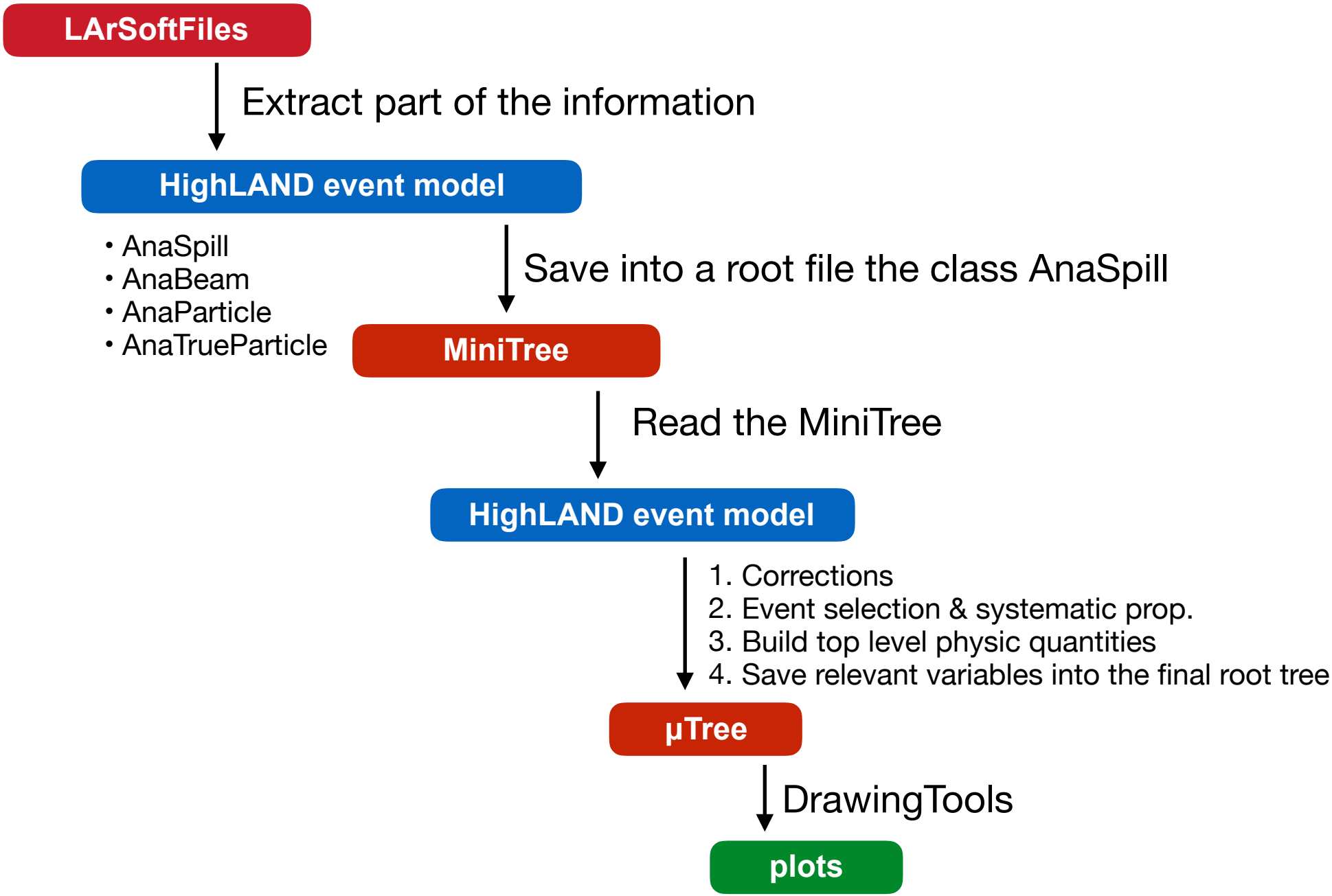
2 seconds for 230 events (a factor 1000)

Summary of data reduction

	LArSoft	MiniTree	Factor
File size	50 GB	20 MB	5000
processing time	2200”	2”	1000

- The MiniTree does not contain yet all info needed for the analysis but I don’t think it will be much larger. Current info:
  - True Particles and True Vertices
  - All reconstructed tracks and showers, and the link to the corresponding True Particle.
  - No hits

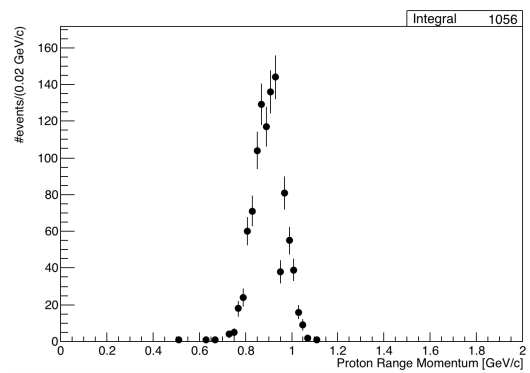
Analysis flow



Data reduction for real DATA

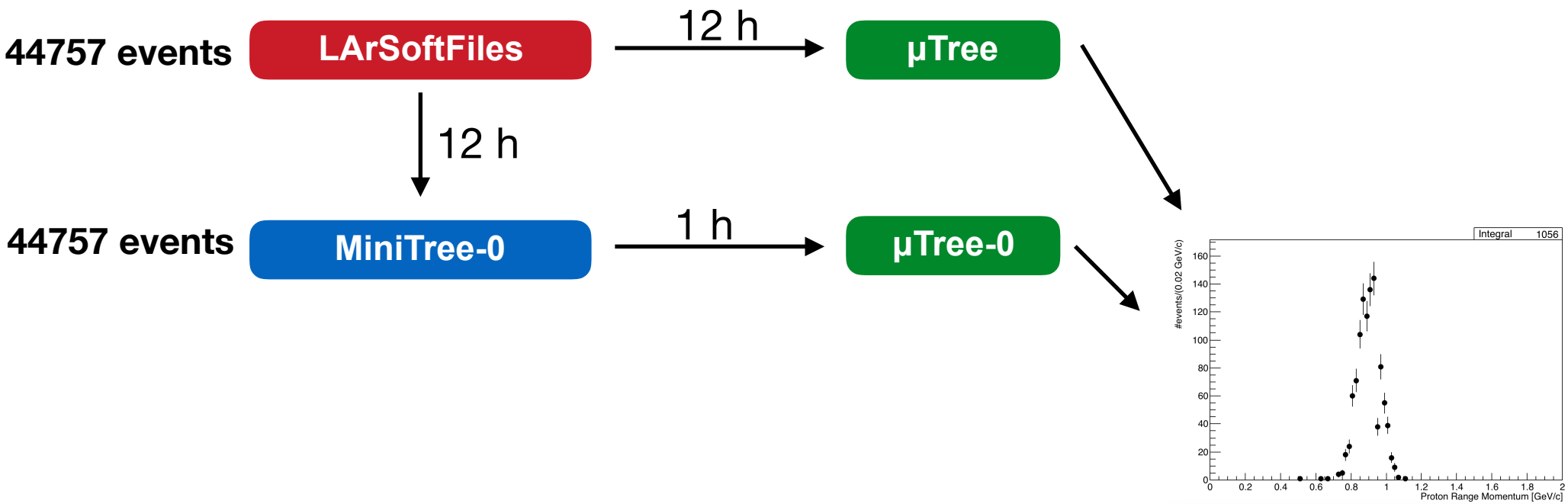
Example for 337 LArSoft files: 1 GeV/c

44757 events → **LArSoftFiles** → 12 h → **μTree**



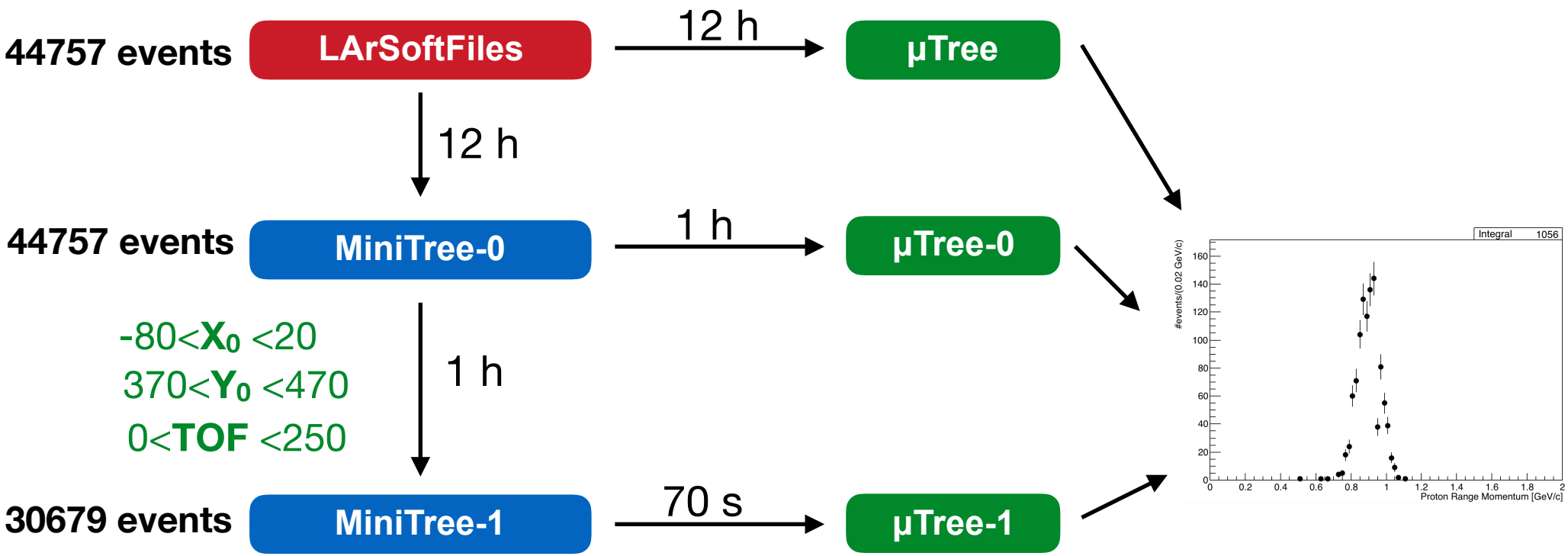
Data reduction for real DATA

Example for 337 LArSoft files: 1 GeV/c



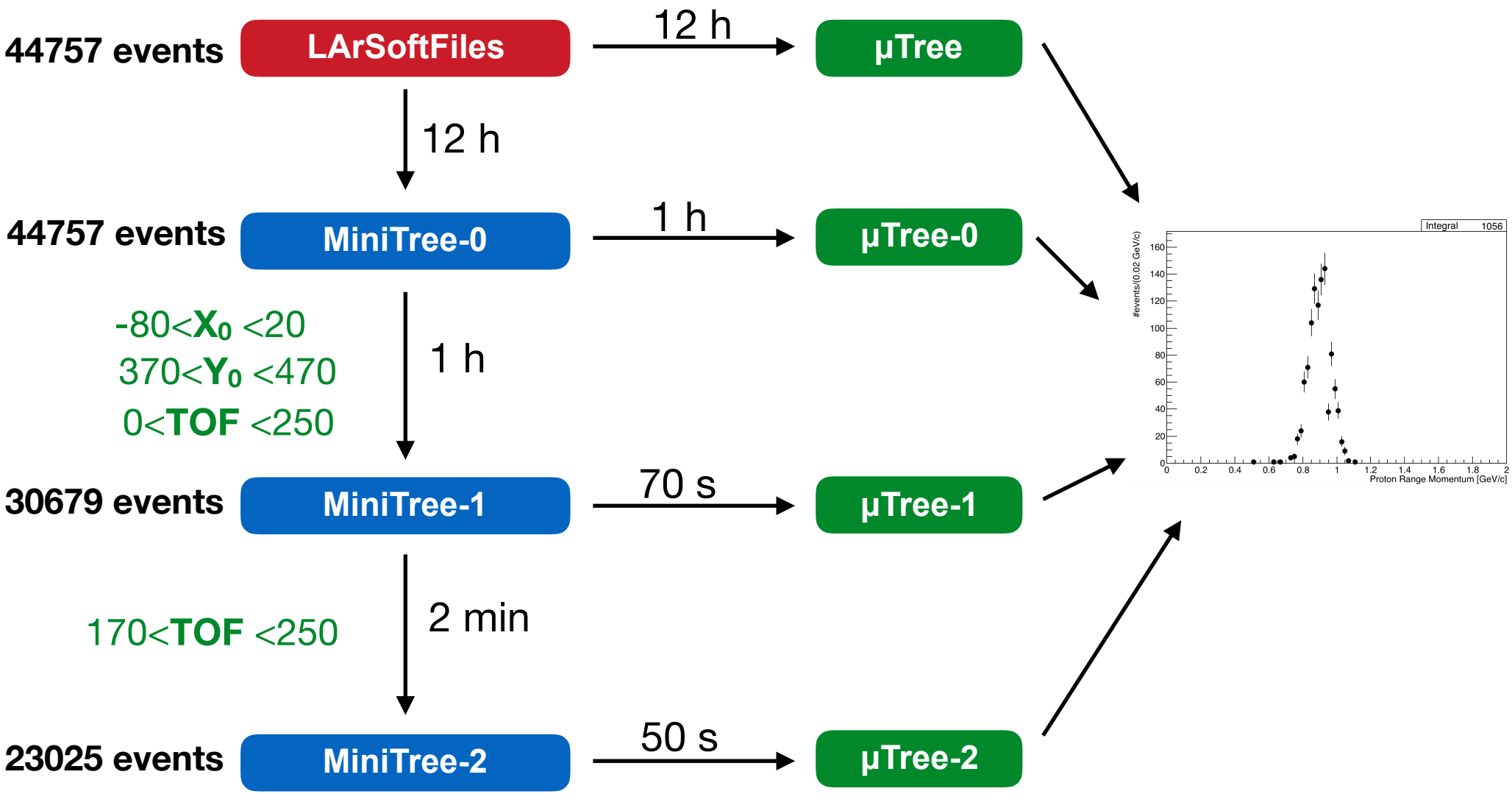
Data reduction for real DATA

Example for 337 LArSoft files: 1 GeV/c



Data reduction for real DATA

Example for 337 LArSoft files: 1 GeV/c



Data reduction for MC

Example for 5225 LArSoft files: 1 GeV/c SCE

Data reduction for MC

Example for 5225 LArSoft files: 1 GeV/c SCE

52250 events
13 TB

LArSoftFiles → 10 days → **μTree**

10 days ↓

52159 events
160 GB

MiniTree-0 → 3 h → **μTree-0**

-80<**X_0** <20
370<**Y_0** <470

3 h ↓

52114 events
36 GB

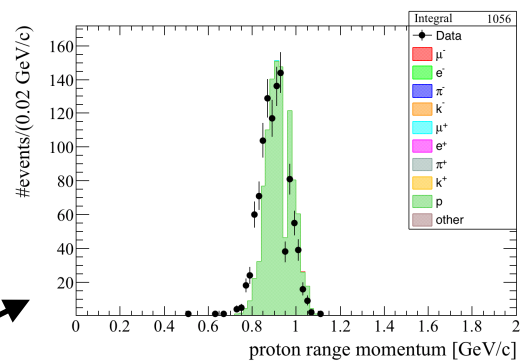
MiniTree-1 → 1 h → **μTree-1**

truth of saved recon particles and all its descendants

1 h ↓

52114 events
8 GB

MiniTree-2 → 3 min → **μTree-2**



Data reduction for MC

Example for 5225 LArSoft files: 1 GeV/c SCE

52250 events
13 TB

LArSoftFiles → 10 days → **μTree**

10 days ↓

52159 events
160 GB

MiniTree-0 → 3 h → **μTree-0**

-80<**X_0** <20
370<**Y_0** <470

3 h ↓

52114 events
36 GB

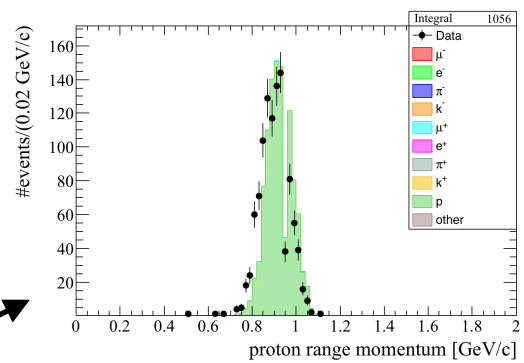
MiniTree-1 → 1 h → **μTree-1**

truth of saved recon particles and all its descendants

1 h ↓

52114 events
8 GB

MiniTree-2 → 3 min → **μTree-2**



In a laptop

Event Selection
Drawing tools

ProtoDUNE example

- An example exists in the git repository: **protoDuneAnalysisExample** package. Documentation exists in redmine. The package contains:
  - stoppingKaonSelection
  - dEdxCorrection, dEdxVariation (systematic)
- Selection of 1 GeV/c beam kaons stopping in the detector
- It was tested using MCC7

Event Selection

- A selection is a collection of steps (cuts or actions)
- Each selection inherits from **SelectionBase**, which has a main mandatory method **DefineSteps**

```
//**********************************************************************
void stoppingKaonSelection::DefineSteps(){
//**********************************************************************

  // Steps must be added in the right order
  // if "true" is added to the constructor of the step,
  // the step sequence is broken if cut is not passed (default is "false")
  AddStep(StepBase::kCut,    "> 0 tracks",         new AtLeastOneTrackCut());
  AddStep(StepBase::kAction, "find true vertex",   new FindTrueVertexAction_proto());
  AddStep(StepBase::kAction, "find main track",    new FindMainTrackAction());
  AddStep(StepBase::kAction, "find vertex",        new FindVertexAction());  // action from duneExampleAnalysis package
  AddStep(StepBase::kCut,    "kaon range",         new KaonRangeCut());
  AddStep(StepBase::kCut,    "> 1 track",          new MoreThanOneTrackCut());
  AddStep(StepBase::kCut,    "PIDA",               new PIDACut());
```

stoppingKaonSelection

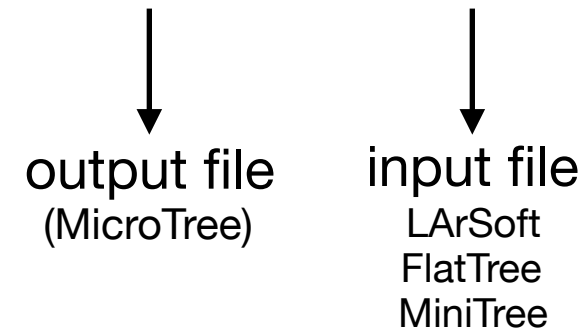
- Each step inherits from **StepBase** and implements the method **Apply**

```
//**************************************************
bool KaonRangeCut::Apply(AnaEventC& event, ToyBoxB& boxB) const{
//**************************************************

  // Cast the ToyBox to the appropriate type
  ToyBoxDUNE& box = *static_cast<ToyBoxDUNE*>(&boxB);
  if (!box.MainTrack) return false;

  Float_t length = static_cast<AnaParticle*>(box.MainTrack)->Length;
  if (fabs(length-200)<10) return true;
  else return false;
}
```

Running and plotting the example

- To run the example

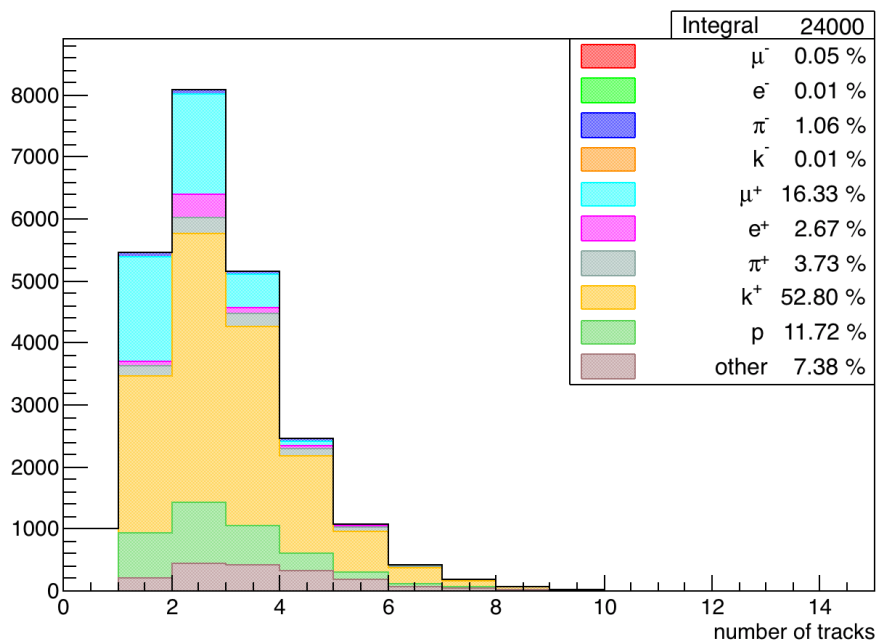
RunProtoDuneAnalysisExample.exe -o *kaon_1gev.root* *input.root*




```
Anselmo-Cerveras-MacBook-Pro-II:cmt acervera$ root -l kaon_1gev.root
root [0]
Attaching file kaon_1gev.root as _file0...
root [1] DrawingTools draw("kaon_1gev.root")
root [2] draw.DumpCuts()
------------------------------------------------------------------------
     Cuts for selection 'stoppingKaonSelection'  with no branches
------------------------------------------------------------------------
  #: index  type       title                    break  branches

  0: -1     cut        > 0 tracks               0      0
  4: -1     cut        kaon range               0      0
  5: -1     cut        > 1 track                0      0
  6: -1     cut        PIDA                     0      0
------------------------------------------------------------------------
```

Track multiplicity

- About 95% of events have at least one reconstructed track
- 1-4 reconstructed tracks is typical
- When >0 tracks, color indicates true particle associated with kaon candidate track in the event (see next)

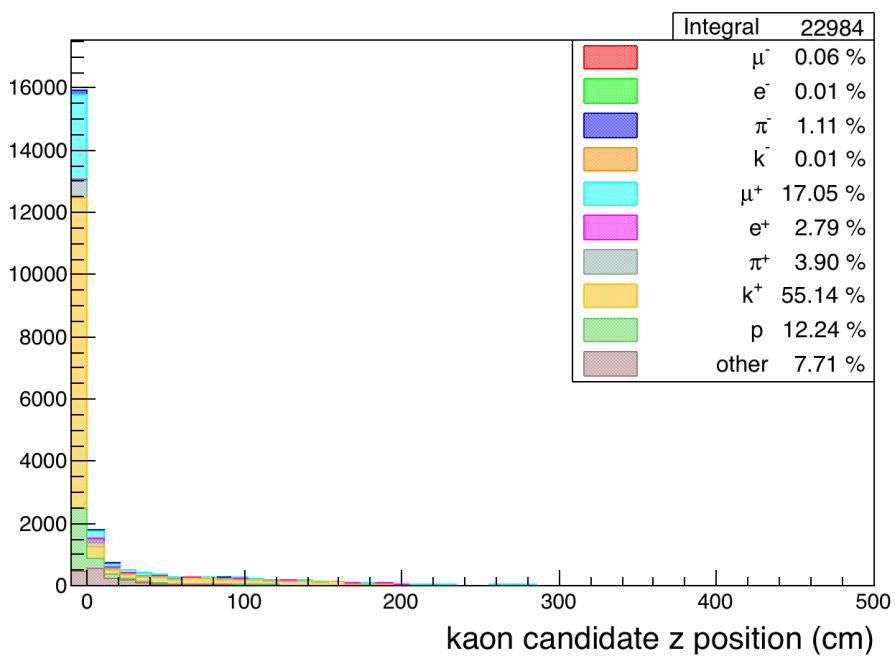


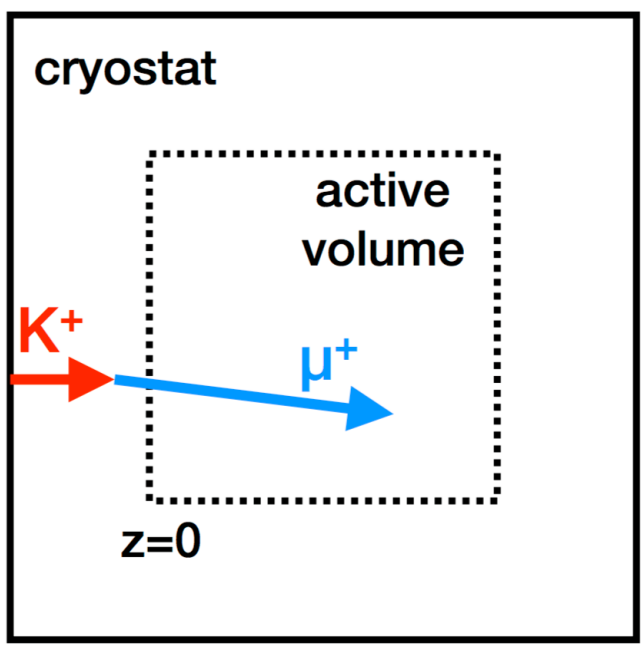
root commands to do the plot

```
draw.SetTitleX("number of tracks");
draw.Draw(default,"ntracks",15,0,15,"particle","accum_level>-1","HIST","DRAWALLMC PUR");
```

Kaon candidate track

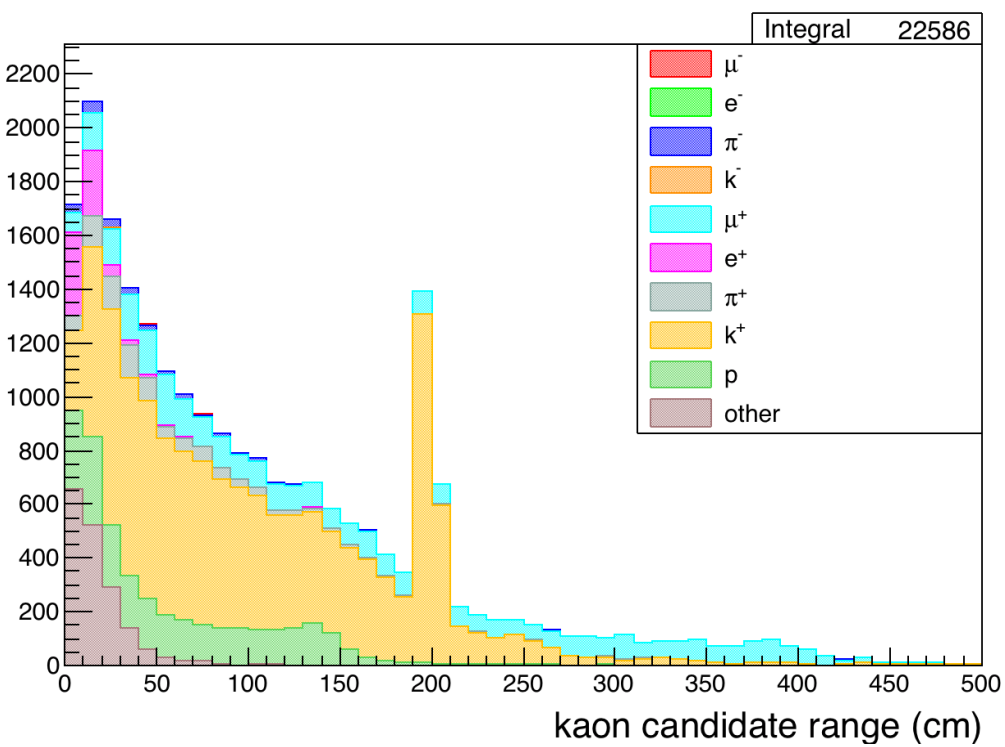
- Kaon candidate track defined as track starting most upstream (lowest z)
- With this definition, kaon candidate associated to true kaon only ~50% of the times
  - Partly because kaon decays or interacts before reaching the active volume (z>0) in ~30% of the simulated events

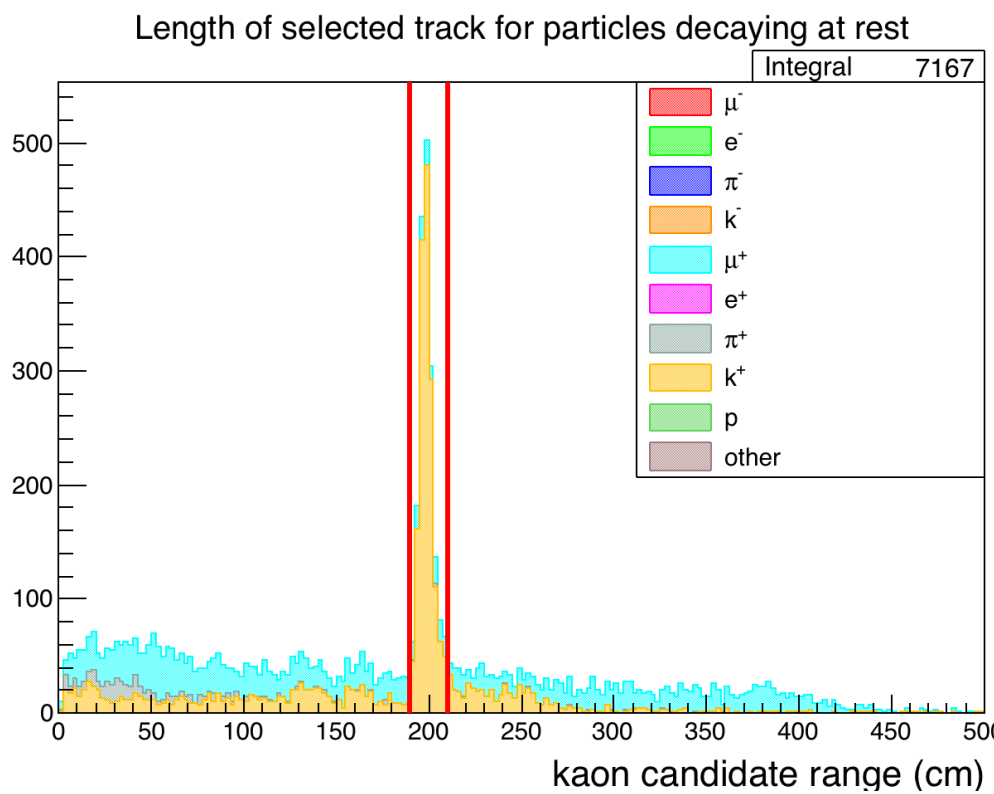




Fraction of stopping kaons and range cut

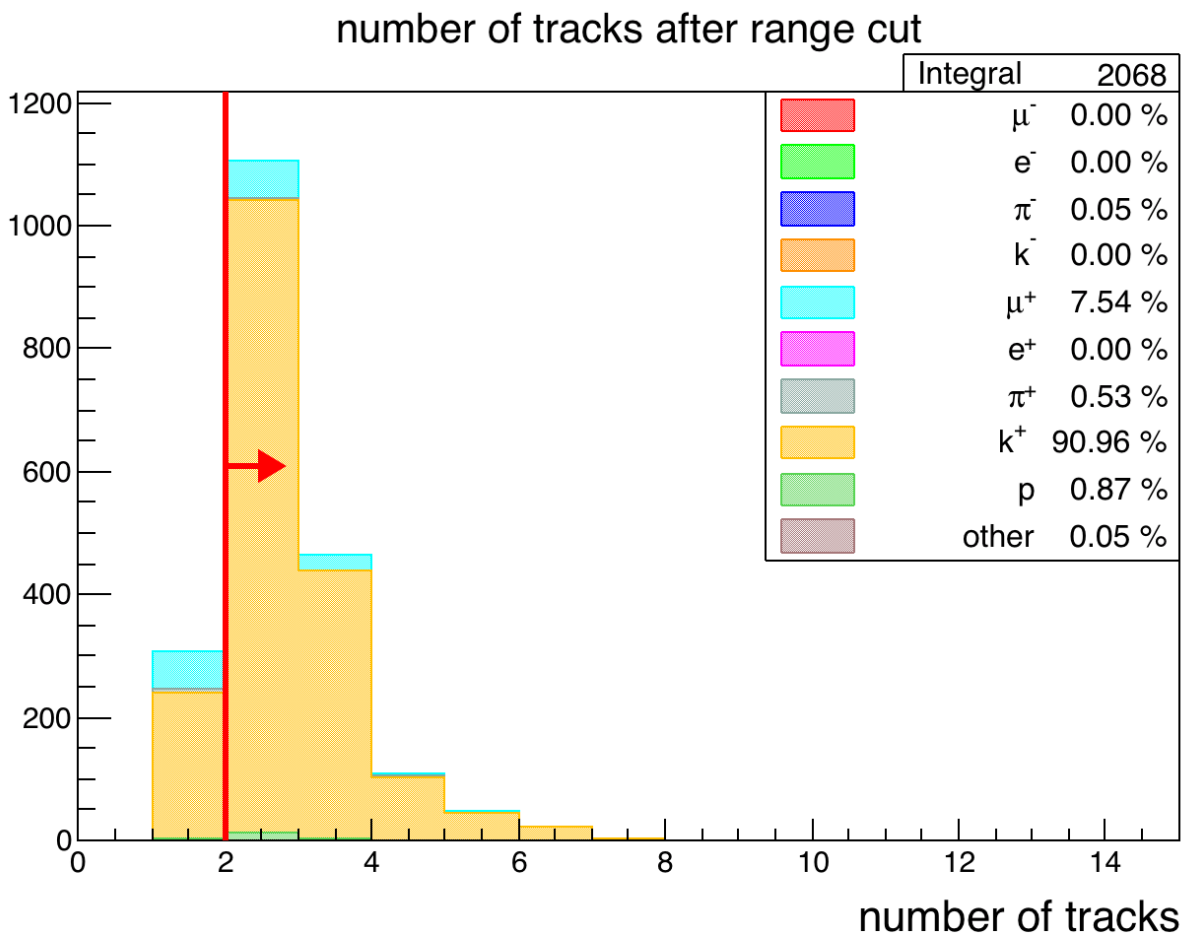
- Only ~10% of the events have a kaon stopping in the active volume
- Require ~200 cm range tracks to select all correctly reconstructed kaons decaying at rest





Multiplicity cut

- In addition to track range, >1 track requirement further improves kaon decay at rest selection
- Reason: tracks from kaon daughters are expected



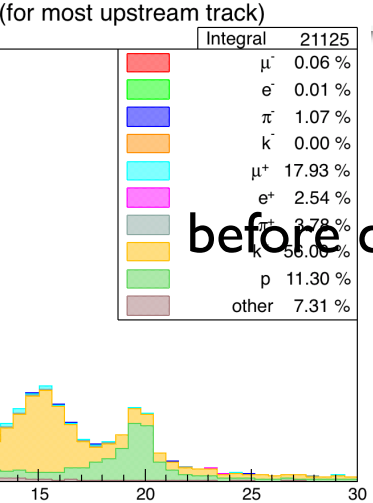
The PIDA variable

Averaged over all hits with residual range R<30

$$PIDA =< A_i >=< (dE/dx)_i R_i^{0.42} >$$

```
//********************************************************************
Float_t anaUtils::ComputePIDA(const AnaParticleB &track) {
//********************************************************************

  Float_t cut=30;

  Float_t PIDA=0;
  Int_t ncontrib=0;
  for (Int_t i=0;i<3;i++){
    for (Int_t j=0;j<track.NHitsPerPlane[i];j++){
      if (track.ResidualRange[i][j]<cut && track.ResidualRange[i][j]>0){
        ncontrib++;
        PIDA += track.dEdx[i][j]*pow(track.ResidualRange[i][j],0.42);
      }
    }
  }
  if (ncontrib>0) PIDA /= ncontrib*1.;

  return PIDA;
```

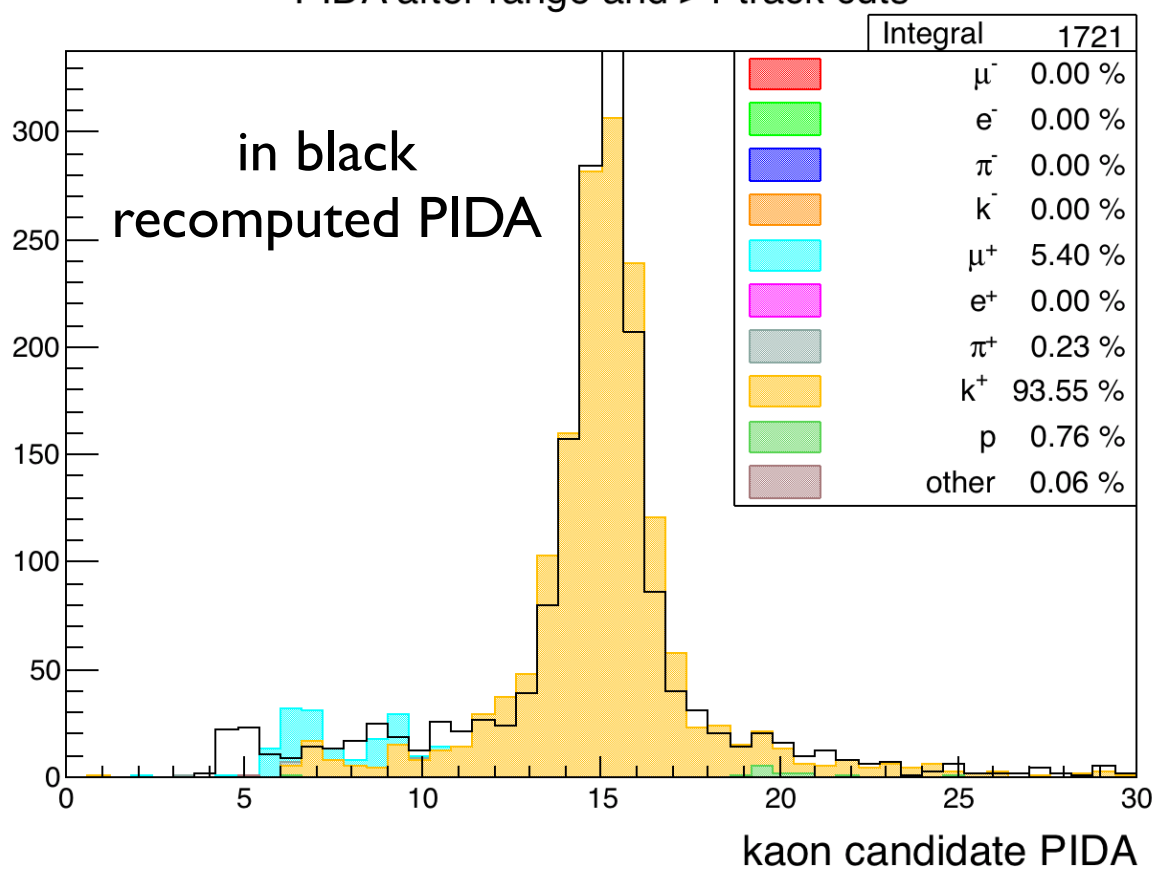
(for most upstream track)



before cuts

kaon candidate PIDA

PIDA after range and >1 track cuts



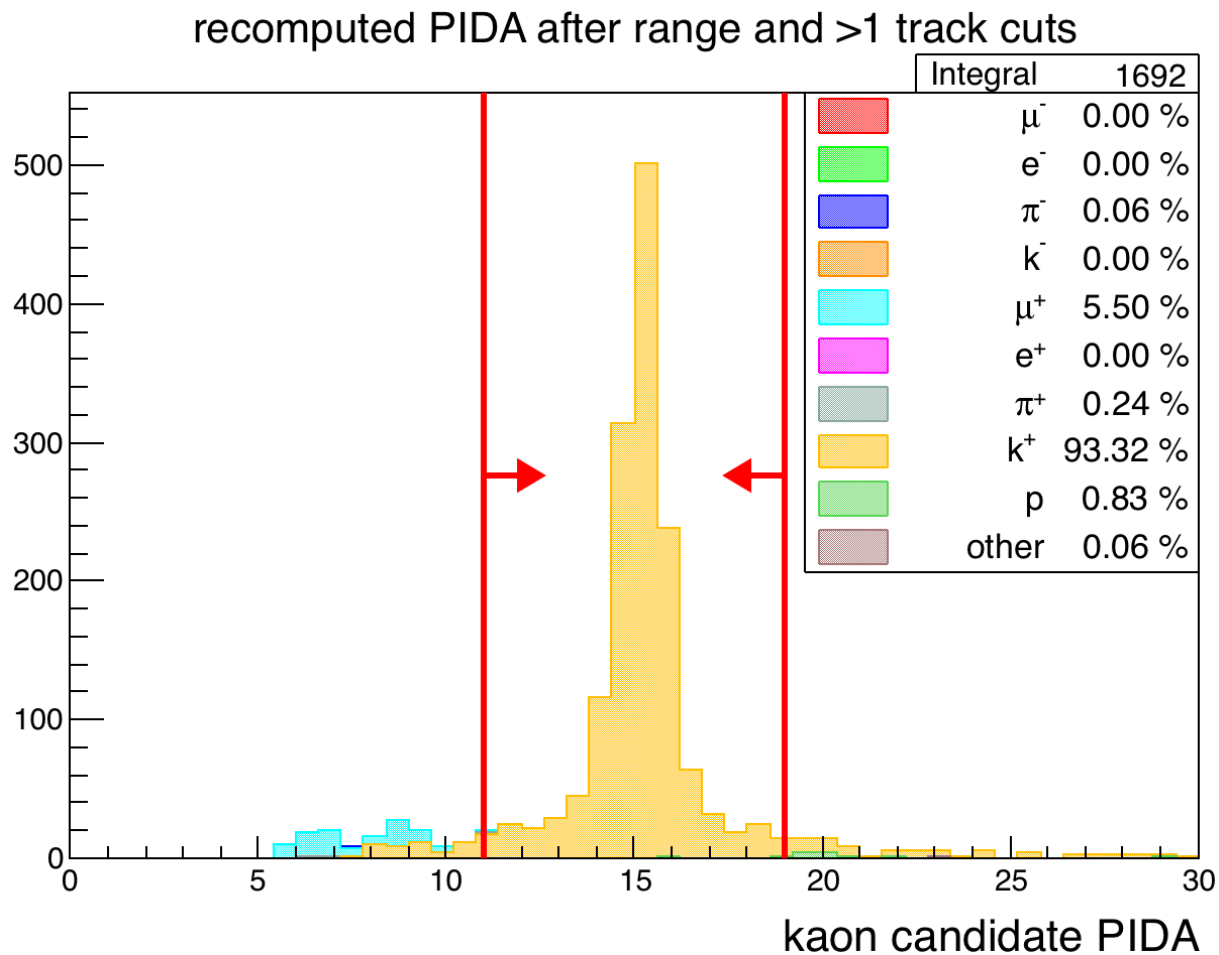
kaon candidate PIDA

The recomputed PIDA is narrower because we have used all 3 wire plane while the one in the AnaTree only use one

The PIDA cut

the code for the cut

```
//************************************************
bool PIDACut::Apply(AnaEventC& event, ToyBoxB& boxB) const{
//************************************************

  (void)event;

  // Cast the ToyBox to the appropriate type
  ToyBoxDUNE& box = *static_cast<ToyBoxDUNE*>(&boxB);
  if (!box.MainTrack) return false;

  Float_t pida = anaUtils::ComputePIDA(*box.MainTrack);
  if (fabs(pida-15.)<4) return true;
  else return false;
}
```

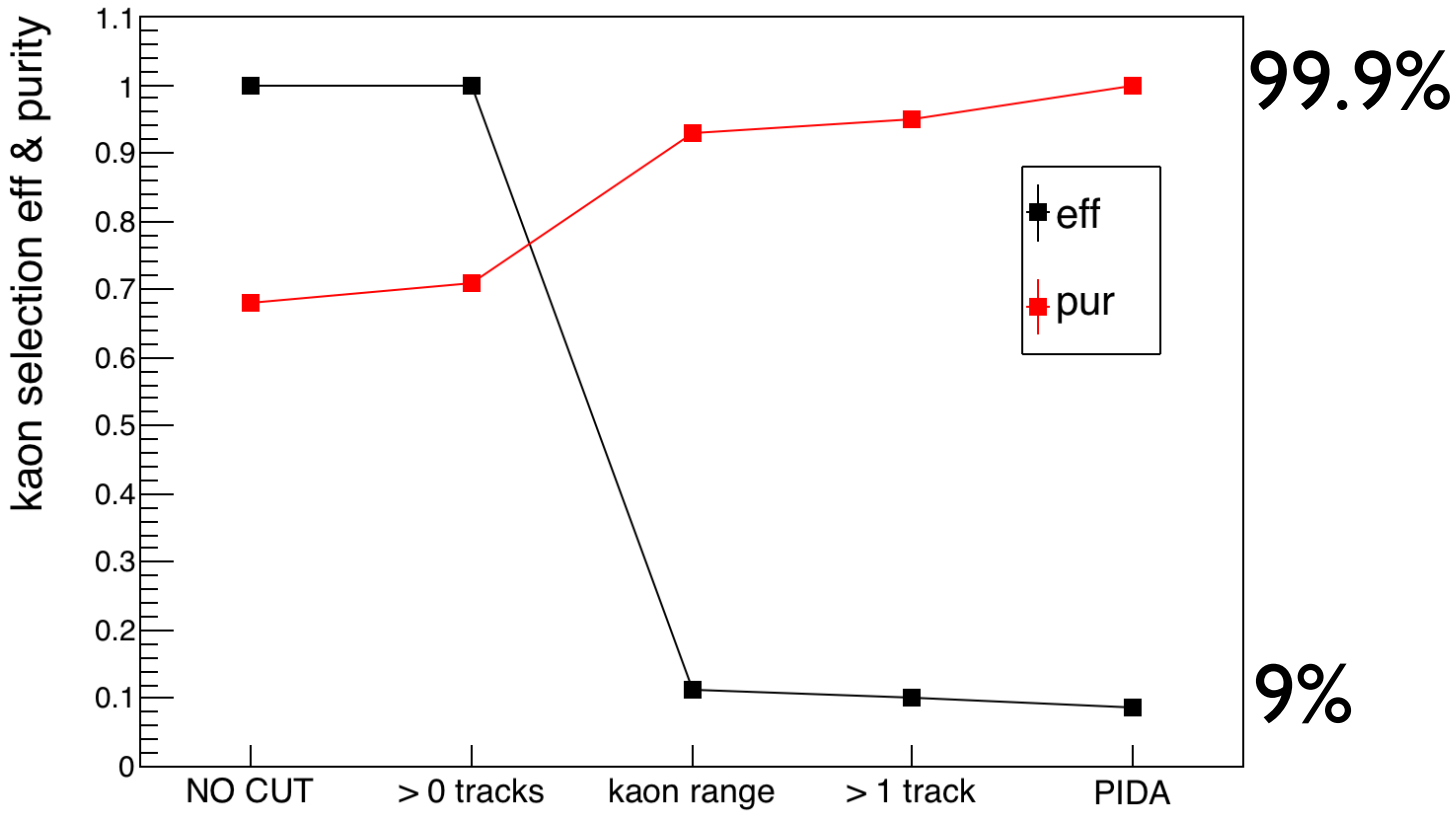


root commands to do the plot

```
draw.SetTitle("recomputed PIDA after range and >1 track cuts");
draw.SetTitleX("kaon candidate PIDA");
draw.Draw(default,"seltrk_pida_raw",50,0,30,"particle","accum_level>2","","PUR");
draw.DrawCutLineVertical(15.-4,true,"r");
draw.DrawCutLineVertical(15.+4,true,"l");
```

accum_level>2 means events passing cut 2
(>0 tracks, range cut, >1 track)

Efficiency and purity



root commands to do the plot

```
// Create a data sample instance with the micro-tree file.
// Needed to plot efficiency (from truth tree) and purity (from default tree) simultaneously
DataSample mc("test.root");

// kaon selection Efficiency and purity after each cut
draw.SetTitleY("kaon selection eff & purity");
draw.DrawEffPurVSCut(mc,"true_signal==1");
```

Reading LarSoft file

Reading the LArSoft file

- Using root TFile::MakeProject the **art** classes headers are recreated

```
recpack:cmt acervera$ ls ../../../LArSoftReader/v0r0/src/v07_02_00/
CRT__Hit.h
CRT__Trigger.h
LArSoftReaderLinkDef.h
LArSoftReaderProjectDict.cxx
LArSoftReaderProjectDict_rdict.pcm
LArSoftReaderProjectHeaders.h
LArSoftReaderProjectInstances.h
LArSoftReaderProjectSource.cxx
LArSoftTreeConverter.cxx
LArSoftTreeConverter.hxx
anab__Calorimetry.h
anab__CosmicTag.h
anab__FeatureVector_4_.h
anab__MVADescription_4_.h
anab__ParticleID.h
anab__T0.h
anab__cosmic_tag_id.h
art__Assns_raw__RawDigit_recob__Hit_void_.h
art__Assns_raw__RawDigit_recob__Wire_void_.h
art__Assns_recob__Cluster_recob__EndPoint2D_unsigned_short_.h
art__Assns_recob__Cluster_recob__EndPoint2D_void_.h
art__Assns_recob__Cluster_recob__Hit_void_.h
art__Assns_recob__Cluster_recob__Vertex_unsigned_short_.h
art__Assns_recob__Cluster_recob__Vertex_void_.h
art__Assns_recob__Hit_recob__SpacePoint_void_.h
art__Assns_recob__OpFlash_recob__OpHit_void_.h
art__Assns_recob__PFParticle_anab__T0_void_.h
art__Assns_recob__PFParticle_larpandoraobj__PFParticleMetadata_void_.h
art__Assns_recob__PFParticle_recob__Cluster_void_.h
art__Assns_recob__PFParticle_recob__PCAxis_void_.h
art__Assns_recob__PFParticle_recob__Shower_void_.h
art__Assns_recob__PFParticle_recob__SpacePoint_void_.h
art__Assns_recob__PFParticle_recob__Track_void_.h
art__Assns_recob__PFParticle_recob__Vertex_void_.h
art__Assns_recob__Shower_recob__Hit_void_.h
art__Assns_recob__Shower_recob__PCAxis_void_.h
art__Assns_recob__SpacePoint_recob__Hit_void_.h
art__Assns_recob__Track_anab__Calorimetry_void_.h
art__Assns_recob__Track_anab__CosmicTag_void_.h
art__Assns_recob__Track_anab__ParticleID_void_.h
art__Assns_recob__Track_anab__T0_void_.h
art__Assns_recob__Track_recob__Hit_recob__TrackHitMeta_.h
art__Assns_recob__Track_recob__Hit_void_.h
art__Assns_recob__Track_recob__SpacePoint_void_.h
art__Assns_recob__Track_recob__Vertex_void_.h
art__Assns_recob__Vertex_recob__Track_void_.h
art__Assns_recob__Wire_recob__Hit_void_.h
art__Assns_sim__AuxDetSimChannel_CRT__Trigger_void_.h
art__Assns_simb__MCTruth_simb__MCParticle_sim__GeneratedParticleInfo_.h
art__Assns_simb__MCTruth_simb__MCParticle_void_.h
art__BranchChildren.h
art__BranchDescription.h
art__BranchKey.h
art__BranchType.h
art__EDProduct.h
art__EventAuxiliary.h
art__EventID.h
art__FileFormatVersion.h
art__FileIndex__Element.h
art__HLTGlobalStatus.h
art__HLTPathStatus.h
art__Hash_2_.h
art__Hash_3_.h
art__Hash_5_.h
art__History.h
art__Parentage.h
art__ProcessConfiguration.h
art__ProcessHistory.h
art__ProductID.h
art__ProductProvenance.h
art__ProductRegistry.h
art__RNGsnapshot.h
art__RefCore.h
art__ResultsAuxiliary.h
art__RunAuxiliary.h
art__RunID.h
art__SubRunAuxiliary.h
art__SubRunID.h
art__Timestamp.h
art__Transient_art__BranchDescription__Transients_.h
art__Transient_art__ProcessHistory__Transients_.h
art__Transient_art__ProductProvenance__Transients_.h
art__TriggerResults.h
art__Wrapper_art__Assns_raw__RawDigit_recob__Hit_void___.h
art__Wrapper_art__Assns_raw__RawDigit_recob__Wire_void___.h
art__Wrapper_art__Assns_recob__Cluster_recob__EndPoint2D_unsigned_short___.h
art__Wrapper_art__Assns_recob__Cluster_recob__Hit_void___.h
art__Wrapper_art__Assns_recob__Cluster_recob__Vertex_unsigned_short___.h
art__Wrapper_art__Assns_recob__Hit_recob__SpacePoint_void___.h
art__Wrapper_art__Assns_recob__OpFlash_recob__OpHit_void___.h
art__Wrapper_art__Assns_recob__PFParticle_anab__T0_void___.h
art__Wrapper_art__Assns_recob__PFParticle_larpandoraobj__PFParticleMetadata_void___.h
art__Wrapper_art__Assns_recob__PFParticle_recob__Cluster_void___.h
art__Wrapper_art__Assns_recob__PFParticle_recob__PCAxis_void___.h
art__Wrapper_art__Assns_recob__PFParticle_recob__Shower_void___.h
art__Wrapper_art__Assns_recob__PFParticle_recob__SpacePoint_void___.h
art__Wrapper_art__Assns_recob__PFParticle_recob__Track_void___.h
art__Wrapper_art__Assns_recob__PFParticle_recob__Vertex_void___.h
art__Wrapper_art__Assns_recob__Shower_recob__Hit_void___.h
art__Wrapper_art__Assns_recob__Shower_recob__PCAxis_void___.h
art__Wrapper_art__Assns_recob__SpacePoint_recob__Hit_void___.h
art__Wrapper_art__Assns_recob__Track_anab__Calorimetry_void___.h
art__Wrapper_art__Assns_recob__Track_anab__CosmicTag_void___.h
art__Wrapper_art__Assns_recob__Track_anab__ParticleID_void___.h
art__Wrapper_art__Assns_recob__Track_anab__T0_void___.h
art__Wrapper_art__Assns_recob__Track_recob__Hit_recob__TrackHitMeta___.h
art__Wrapper_art__Assns_recob__Track_recob__Hit_void___.h
art__Wrapper_art__Assns_recob__Track_recob__SpacePoint_void___.h
art__Wrapper_art__Assns_recob__Track_recob__Vertex_void___.h
art__Wrapper_art__Assns_recob__Vertex_recob__Track_void___.h
art__Wrapper_art__Assns_recob__Wire_recob__Hit_void___.h
art__Wrapper_art__Assns_sim__AuxDetSimChannel_CRT__Trigger_void___.h
art__Wrapper_art__Assns_simb__MCTruth_simb__MCParticle_sim__GeneratedParticleInfo___.h
art__Wrapper_art__TriggerResults_.h
art__Wrapper_vector_CRT__Trigger___.h
art__Wrapper_vector_anab__Calorimetry___.h
art__Wrapper_vector_anab__CosmicTag___.h
art__Wrapper_vector_anab__FeatureVector_4_____.h
art__Wrapper_vector_anab__MVADescription_4_____.h
art__Wrapper_vector_anab__ParticleID___.h
art__Wrapper_vector_anab__T0___.h
art__Wrapper_vector_art__RNGsnapshot___.h
art__Wrapper_vector_larpandoraobj__PFParticleMetadata___.h
art__Wrapper_vector_raw__OpDetWaveform___.h
art__Wrapper_vector_raw__RawDigit___.h
art__Wrapper_vector_recob__Cluster___.h
art__Wrapper_vector_recob__EndPoint2D___.h
art__Wrapper_vector_recob__Hit___.h
art__Wrapper_vector_recob__OpFlash___.h
art__Wrapper_vector_recob__OpHit___.h
art__Wrapper_vector_recob__PCAxis___.h
art__Wrapper_vector_recob__PFParticle___.h
art__Wrapper_vector_recob__PointCharge___.h
art__Wrapper_vector_recob__Shower___.h
art__Wrapper_vector_recob__SpacePoint___.h
art__Wrapper_vector_recob__Track___.h
art__Wrapper_vector_recob__Vertex___.h
art__Wrapper_vector_recob__Wire___.h
art__Wrapper_vector_sim__AuxDetSimChannel___.h
art__Wrapper_vector_sim__OpDetBacktrackerRecord___.h
art__Wrapper_vector_sim__OpDetDivRec___.h
art__Wrapper_vector_sim__SimChannel___.h
art__Wrapper_vector_sim__SimPhotonsLite___.h
art__Wrapper_vector_simb__MCParticle___.h
art__Wrapper_vector_simb__MCTruth___.h
art__detail__AssnsBase.h
fhicl__ParameterSetID.h
geo__CryostatID.h
geo__PlaneID.h
geo__TPCID.h
geo__WireID.h
geo___plane_proj.h
geo___plane_sigtype.h
lar__range_t_unsigned_long_.h
lar__sparse_vector_float_.h
larpandoraobj__PFParticleMetadata.h
raw__OpDetWaveform.h
raw__RawDigit.h
raw___compress.h
recob__Cluster.h
recob__EndPoint2D.h
recob__Hit.h
recob__OpFlash.h
recob__OpHit.h
recob__PCAxis.h
recob__PFParticle.h
recob__PointCharge.h
recob__Shower.h
recob__SpacePoint.h
recob__Track.h
recob__TrackHitMeta.h
recob__TrackTrajectory.h
recob__Trajectory.h
recob__TrajectoryPointFlags.h
recob__Vertex.h
recob__Wire.h
sim__AuxDetIDE.h
sim__AuxDetSimChannel.h
sim__Chan_Phot.h
sim__GeneratedParticleInfo.h
sim__IDE.h
sim__OpDetBacktrackerRecord.h
sim__OpDetDivRec.h
sim__OpDet_Time_Chans.h
sim__SDP.h
sim__SimChannel.h
sim__SimPhotonsLite.h
simb__MCNeutrino.h
simb__MCParticle.h
simb__MCTrajectory.h
simb__MCTruth.h
simb___ev_origin.h
util__flags__BitMask_unsigned_int_.h
util__flags__Bits_t_unsigned_int_.h
util__flags__FlagSet_32_unsigned_int_.h
```

- So we don't need **art** nor **LArSoft** to read the LArSoft file

Reading the LArSoft file

- We read the “Events” tree in a LarSoft file

```
// General event info
eventsTree->SetBranchAddress("EventAuxiliary", &EventInfo);

// Reconstructed tracks
eventsTree->SetBranchAddress("recob::Tracks_pmtrackdc__Reco.", &Tracks);

// MC particles
eventsTree->SetBranchAddress("simb::MCParticles_largeant__G4.", &MCParticles);

// MC neutrinos
eventsTree->SetBranchAddress("simb::MCTruths_generator__GenieGen.", &MCNeutrinos);

// Reconstructed hits
eventsTree->SetBranchAddress("recob::Hits_lineclusterdc__Reco.", &Hits);

// Association between reconstructed hits and tracks
eventsTree->SetBranchAddress("recob::Hitrecob::Trackvoidart::Assns_pmtrack__Reco.", &Hits_Tracks);

// Channels
eventsTree->SetBranchAddress("sim::SimChannels_largeant__G4.", &SimChannels);
```

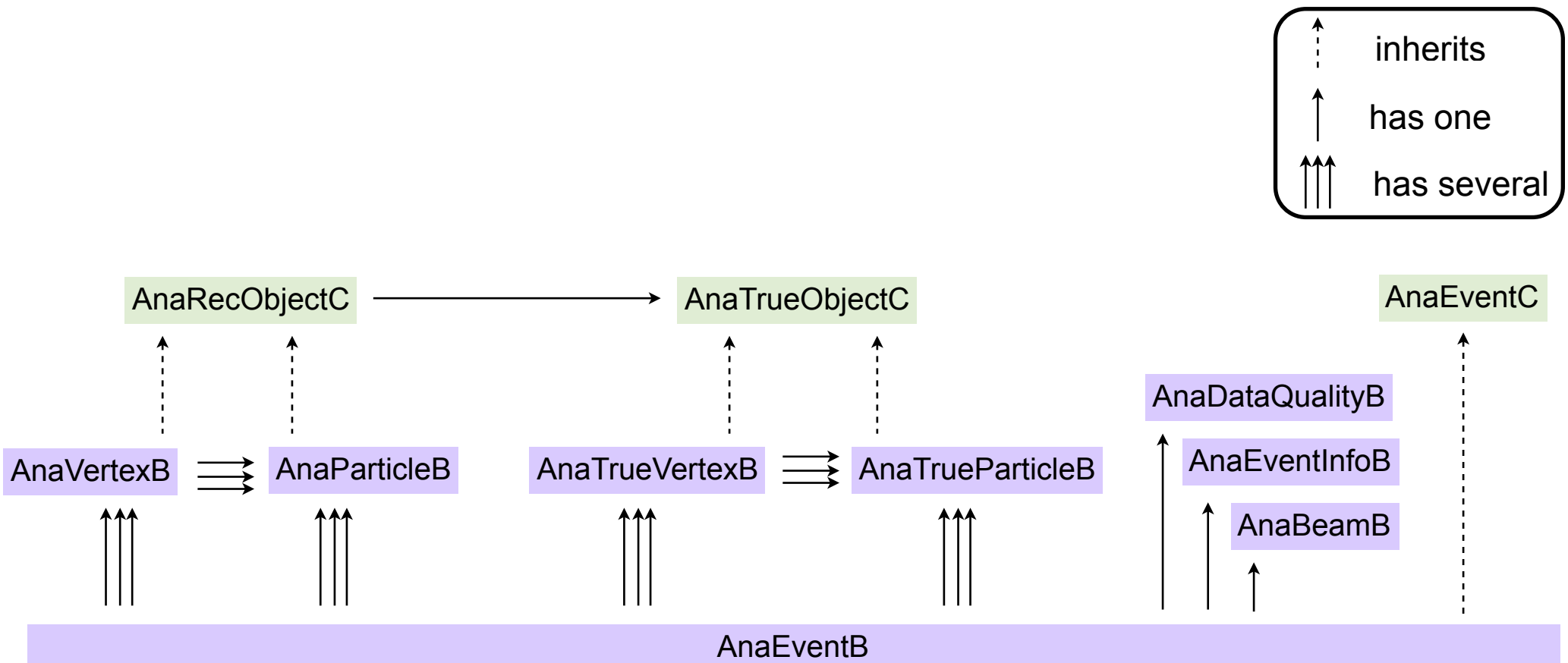
- Disable all branches we are not interested in to gain in speed

```
//-------------------- Disable the unnecessary branches ---
eventsTree->SetBranchStatus("art::*",0);
eventsTree->SetBranchStatus("sim::Beam*",0);
eventsTree->SetBranchStatus("sim::AuxDet*",0);
eventsTree->SetBranchStatus("sim::SimPhoton*",0);

eventsTree->SetBranchStatus("*shower*",0);

eventsTree->SetBranchStatus("raw*",0);
eventsTree->SetBranchStatus("anab*",0);
eventsTree->SetBranchStatus("recob::Wires*",0);
eventsTree->SetBranchStatus("recob::Vertexs*",0);
eventsTree->SetBranchStatus("recob::Space*",0);
eventsTree->SetBranchStatus("recob::Trackrecob*",0);
eventsTree->SetBranchStatus("recob::Tracks_emshower*",0);
```

```
eventsTree->SetBranchStatus("recob::Tracks_pmtrack_*",0);
eventsTree->SetBranchStatus("recob::Shower*",0);
eventsTree->SetBranchStatus("recob::Op*",0);
eventsTree->SetBranchStatus("recob::Hits_dcheat*",0);
eventsTree->SetBranchStatus("recob::Hits_gaushit*",0);
eventsTree->SetBranchStatus("recob::Hits_hitfd*",0);
eventsTree->SetBranchStatus("recob::Hitrecob::Space*",0);
eventsTree->SetBranchStatus("recob::Hitrecob::Wire*",0);
eventsTree->SetBranchStatus("recob::End*",0);
eventsTree->SetBranchStatus("recob::Cluster*",0);
//  eventsTree->SetBranchStatus("simb::MCTruths*",0);
eventsTree->SetBranchStatus("simb::MCFlux*",0);
eventsTree->SetBranchStatus("simb::GTruth*",0);
eventsTree->SetBranchStatus("simb::MCParticlesimb*",0);
```

HighLAND event model

- LArSoft info is extracted and saved into the HighLAND event model
- This is the current event model, which can be easily modified



Track and shower info

- This is the method to fill the AnaParticle's (track/shower)

```
//********************************************************************************
void LArSoftTreeConverter::FillParticleInfo(std::vector<AnaTrueParticleB*>& trueParticles, Int_t itrk, AnaParticle* part){
//********************************************************************************

  // The  ID of the particle (track or shower)
  part->UniqueID = Tracks->obj[itrk].fID;

  // General Particle ofinfo
  for (UInt_t j=0;j<3;j++){
    part->PositionStart[j]= Tracks->obj[itrk].fXYZ[0][j];
    part->DirectionStart[j]= Tracks->obj[itrk].fDir[0][j];
    part->PositionEnd[j]= Tracks->obj[itrk].fXYZ[Tracks->obj[itrk].fXYZ.size()-1][j];
    part->DirectionEnd[j]= Tracks->obj[itrk].fDir[Tracks->obj[itrk].fXYZ.size()-1][j];
  }

  // The number of hits
  part->NHits = Tracks->obj[itrk].fXYZ.size();
  SubDetId::SetDetectorUsed(part->Detector , SubDetId::kSubdet1_1);
  SubDetId::SetDetectorSystemFields(part->Detector);

  // Compute the track length
  part->Length = ComputeTrackLength(Tracks->obj[itrk]);

  // Compute the average dQdx of the reconstructed particle
  Int_t nsamples=0;
  for (UInt_t i=0;i<Tracks->obj[itrk].fdQdx.size();i++){
    for (UInt_t j=0;j<Tracks->obj[itrk].fdQdx[i].size();j++){
      part->AveragedEdx += Tracks->obj[itrk].fdQdx[i][j];
      nsamples++;
    }
  }
  part->AveragedEdx /= (Float_t)nsamples;

  // Associate a TrueObject to this Particle
  part->TrueObject = FindTrueParticle(itrk, trueParticles);
}
```

There is a method in LArSoft to do this. We had to reproduce that method inside HighLAND

- reco_beam_
- reco_daughter_ : daughter tracks from beam particle
    - reco_daughter_true_byE_
    - reco_daughter_true_byHits
    - reco_daughter_PFP_true_byHits_
    - reco_daughter_allTrack_
    - reco_daughter_allShower_
- reco_daughter_shower_true_byE
- reco_daughter_shower_true_byHits_
- reco_daughter_
- true_beam_
- reco_beam_
- true_daughter

reco beam

```
// Get the reconstructed PFParticle tagged as beam by Pandora
const recob::PFParticle* particle = beamParticles.at(0);

// Determine if the beam particle is track-like or shower-like
const recob::Track* thisTrack = pfpUtil.GetPFParticleTrack(*particle,evt,fPFParticleTag,fTrackerTag);
const recob::Shower* thisShower = pfpUtil.GetPFParticleShower(*particle,evt,fPFParticleTag,fShowerTag);
const simb::MCParticle* trueParticle = 0x0;
```

```
reco_beam_startX = thisTrack->Trajectory().Start().X();
reco_beam_startY = thisTrack->Trajectory().Start().Y();
reco_beam_startZ = thisTrack->Trajectory().Start().Z();
reco_beam_endX = thisTrack->Trajectory().End().X();
reco_beam_endY = thisTrack->Trajectory().End().Y();
reco_beam_endZ = thisTrack->Trajectory().End().Z();
```

```
protoana::MCParticleSharedHits beam_match = truthUtil.GetMCParticleByHits( *thisTrack, evt, fTrackerTag, fHitTag );
if( beam_match.particle ){
  //Check that this is the correct true particle
  if( beam_match.particle->TrackId() == true_beam_particle->TrackId() ){
    reco_beam_true_byHits_matched = true;
  }

  reco_beam_true_byHits_PDG = beam_match.particle->PdgCode();
```

reco beam

```
fTree->Branch("reco_beam_type", &reco_beam_type);
fTree->Branch("reco_beam_startX", &reco_beam_startX);
fTree->Branch("reco_beam_startY", &reco_beam_startY);
fTree->Branch("reco_beam_startZ", &reco_beam_startZ);
fTree->Branch("reco_beam_endX", &reco_beam_endX);
fTree->Branch("reco_beam_endY", &reco_beam_endY);
fTree->Branch("reco_beam_endZ", &reco_beam_endZ);
fTree->Branch("reco_beam_len", &reco_beam_len);
fTree->Branch("reco_beam_trackDirX", &reco_beam_trackDirX);
fTree->Branch("reco_beam_trackDirY", &reco_beam_trackDirY);
fTree->Branch("reco_beam_trackDirZ", &reco_beam_trackDirZ);
fTree->Branch("reco_beam_trackEndDirX", &reco_beam_trackEndDirX);
fTree->Branch("reco_beam_trackEndDirY", &reco_beam_trackEndDirY);
fTree->Branch("reco_beam_trackEndDirZ", &reco_beam_trackEndDirZ);
fTree->Branch("reco_beam_vtxX", &reco_beam_vtxX);
fTree->Branch("reco_beam_vtxY", &reco_beam_vtxY);
fTree->Branch("reco_beam_vtxZ", &reco_beam_vtxZ);
fTree->Branch("reco_beam_trackID", &reco_beam_trackID);
fTree->Branch("reco_beam_dQdX", &reco_beam_dQdX);
fTree->Branch("reco_beam_dEdX", &reco_beam_dEdX);
fTree->Branch("reco_beam_calibrated_dEdX", &reco_beam_calibrated_dEdX);
fTree->Branch("reco_beam_resRange", &reco_beam_resRange);
fTree->Branch("reco_beam_TrkPitch", &reco_beam_TrkPitch);
fTree->Branch("reco_beam_calo_wire", &reco_beam_calo_wire);
fTree->Branch("reco_beam_calo_tick", &reco_beam_calo_tick);
fTree->Branch("reco_beam_nTrackDaughters", &reco_beam_nTrackDaughters);
fTree->Branch("reco_beam_nShowerDaughters", &reco_beam_nShowerDaughters);
fTree->Branch("reco_beam_flipped", &reco_beam_flipped);
fTree->Branch("reco_beam_passes_beam_cuts", &reco_beam_passes_beam_cuts);
fTree->Branch("reco_beam_vertex_slice", &reco_beam_vertex_slice);
fTree->Branch("reco_beam_vertex_dRs", &reco_beam_vertex_dRs);
fTree->Branch("reco_beam_vertex_hits_slices", &reco_beam_vertex_hits_slices);
fTree->Branch("reco_beam_true_byE_endProcess", &reco_beam_true_byE_endProcess);
fTree->Branch("reco_beam_true_byE_process", &reco_beam_true_byE_process);
fTree->Branch("reco_beam_true_byE_origin", &reco_beam_true_byE_origin);
fTree->Branch("reco_beam_true_byE_PDG", &reco_beam_true_byE_PDG);
fTree->Branch("reco_beam_true_byE_ID", &reco_beam_true_byE_ID);
fTree->Branch("reco_beam_true_byHits_endProcess", &reco_beam_true_byHits_endProcess);
fTree->Branch("reco_beam_true_byHits_process", &reco_beam_true_byHits_process);
fTree->Branch("reco_beam_true_byHits_origin", &reco_beam_true_byHits_origin);
fTree->Branch("reco_beam_true_byHits_PDG", &reco_beam_true_byHits_PDG);
fTree->Branch("reco_beam_true_byHits_ID", &reco_beam_true_byHits_ID);

fTree->Branch("reco_beam_true_byE_matched", &reco_beam_true_byE_matched);
fTree->Branch("reco_beam_true_byHits_matched", &reco_beam_true_byHits_matched);
fTree->Branch("reco_beam_true_byHits_purity", &reco_beam_true_byHits_purity);

fTree->Branch("reco_beam_Chi2_proton", &reco_beam_Chi2_proton);
fTree->Branch("reco_beam_Chi2_ndof", &reco_beam_Chi2_ndof);

fTree->Branch("reco_beam_cosmic_candidate_lower_hits", &reco_beam_cosmic_candidate_lower_hits);
fTree->Branch("reco_beam_cosmic_candidate_upper_hits", &reco_beam_cosmic_candidate_upper_hits);
fTree->Branch("reco_beam_cosmic_candidate_ID", &reco_beam_cosmic_candidate_ID)
fTree->Branch("reco_beam_true_byE_endPx", &reco_beam_true_byE_endPx);
fTree->Branch("reco_beam_true_byE_endPy", &reco_beam_true_byE_endPy);
fTree->Branch("reco_beam_true_byE_endPz", &reco_beam_true_byE_endPz);
fTree->Branch("reco_beam_true_byE_endE", &reco_beam_true_byE_endE);
fTree->Branch("reco_beam_true_byE_endP", &reco_beam_true_byE_endP);

fTree->Branch("reco_beam_true_byE_startPx", &reco_beam_true_byE_startPx);
fTree->Branch("reco_beam_true_byE_startPy", &reco_beam_true_byE_startPy);
fTree->Branch("reco_beam_true_byE_startPz", &reco_beam_true_byE_startPz);
fTree->Branch("reco_beam_true_byE_startE", &reco_beam_true_byE_startE);
fTree->Branch("reco_beam_true_byE_startP", &reco_beam_true_byE_startP);


fTree->Branch("reco_beam_true_byHits_endPx", &reco_beam_true_byHits_endPx);
fTree->Branch("reco_beam_true_byHits_endPy", &reco_beam_true_byHits_endPy);
fTree->Branch("reco_beam_true_byHits_endPz", &reco_beam_true_byHits_endPz);
fTree->Branch("reco_beam_true_byHits_endE", &reco_beam_true_byHits_endE);
fTree->Branch("reco_beam_true_byHits_endP", &reco_beam_true_byHits_endP);

fTree->Branch("reco_beam_true_byHits_startPx", &reco_beam_true_byHits_startPx);
fTree->Branch("reco_beam_true_byHits_startPy", &reco_beam_true_byHits_startPy);
fTree->Branch("reco_beam_true_byHits_startPz", &reco_beam_true_byHits_startPz);
fTree->Branch("reco_beam_true_byHits_startE", &reco_beam_true_byHits_startE);
fTree->Branch("reco_beam_true_byHits_startP", &reco_beam_true_byHits_startP);

  fTree->Branch( "reco_beam_spacePts_X", &reco_beam_spacePts_X );
  fTree->Branch( "reco_beam_spacePts_Y", &reco_beam_spacePts_Y );
  fTree->Branch( "reco_beam_spacePts_Z", &reco_beam_spacePts_Z );
;
```

true beam

```
// This gets the true beam particle that generated the event
const simb::MCParticle* true_beam_particle = 0x0;
if( !evt.isRealData() ){
  auto mcTruths = evt.getValidHandle<std::vector<simb::MCTruth>>(fGeneratorTag);
  true_beam_particle = truthUtil.GetGeantGoodParticle((*mcTruths)[0],evt);
  if( !true_beam_particle ){
    std::cout << "No true beam particle" << std::endl;
    return;
  }
}
```

true beam

```
fTree->Branch("true_beam_PDG", &true_beam_PDG);
fTree->Branch("true_beam_ID", &true_beam_ID);
fTree->Branch("true_beam_endProcess", &true_beam_endProcess);
fTree->Branch("true_beam_endX", &true_beam_endX);
fTree->Branch("true_beam_endY", &true_beam_endY);
fTree->Branch("true_beam_endZ", &true_beam_endZ);
fTree->Branch("true_beam_startX", &true_beam_startX);
fTree->Branch("true_beam_startY", &true_beam_startY);
fTree->Branch("true_beam_startZ", &true_beam_startZ);

fTree->Branch("true_beam_startPx", &true_beam_startPx);
fTree->Branch("true_beam_startPy", &true_beam_startPy);
fTree->Branch("true_beam_startPz", &true_beam_startPz);
fTree->Branch("true_beam_startP", &true_beam_startP);

fTree->Branch("true_beam_endPx", &true_beam_endPx);
fTree->Branch("true_beam_endPy", &true_beam_endPy);
fTree->Branch("true_beam_endPz", &true_beam_endPz);
fTree->Branch("true_beam_endP", &true_beam_endP);

fTree->Branch("true_beam_startDirX", &true_beam_startDirX);
fTree->Branch("true_beam_startDirY", &true_beam_startDirY);
fTree->Branch("true_beam_startDirZ", &true_beam_startDirZ);

fTree->Branch("true_beam_nElasticScatters", &true_beam_nElasticScatters);
fTree->Branch("true_beam_elastic_costheta", &true_beam_elastic_costheta);
fTree->Branch("true_beam_elastic_X", &true_beam_elastic_X);
fTree->Branch("true_beam_elastic_Y", &true_beam_elastic_Y);
fTree->Branch("true_beam_elastic_Z", &true_beam_elastic_Z);
fTree->Branch("true_beam_IDE_totalDep",    &true_beam_IDE_totalDep);
fTree->Branch("true_beam_IDE_found_in_recoVtx",    &true_beam_IDE_found_in_recoVtx);

fTree->Branch("true_beam_nHits", &true_beam_nHits);
fTree->Branch("true_beam_reco_byHits_PFP_ID", &true_beam_reco_byHits_PFP_ID);
fTree->Branch("true_beam_reco_byHits_PFP_nHits", &true_beam_reco_byHits_PFP_nHits);
fTree->Branch("true_beam_reco_byHits_allTrack_ID", &true_beam_reco_byHits_allTrack_ID);
```

```
fTree->Branch("true_beam_daughter_PDG", &true_beam_daughter_PDG);
fTree->Branch("true_beam_daughter_ID", &true_beam_daughter_ID);
fTree->Branch("true_beam_daughter_len", &true_beam_daughter_len);
fTree->Branch("true_beam_daughter_startX", &true_beam_daughter_startX);
fTree->Branch("true_beam_daughter_startY", &true_beam_daughter_startY);
fTree->Branch("true_beam_daughter_startZ", &true_beam_daughter_startZ);
fTree->Branch("true_beam_daughter_startPx", &true_beam_daughter_startPx);
fTree->Branch("true_beam_daughter_startPy", &true_beam_daughter_startPy);
fTree->Branch("true_beam_daughter_startPz", &true_beam_daughter_startPz);
fTree->Branch("true_beam_daughter_startP", &true_beam_daughter_startP);
fTree->Branch("true_beam_daughter_endX", &true_beam_daughter_endX);
fTree->Branch("true_beam_daughter_endY", &true_beam_daughter_endY);
fTree->Branch("true_beam_daughter_endZ", &true_beam_daughter_endZ);
fTree->Branch("true_beam_daughter_Process", &true_beam_daughter_Process);
fTree->Branch("true_beam_daughter_endProcess", &true_beam_daughter_endProcess);
fTree->Branch("true_beam_daughter_nHits", &true_beam_daughter_nHits);

fTree->Branch("true_beam_daughter_reco_byHits_PFP_ID", &true_beam_daughter_reco_byHits_PFP_ID);
fTree->Branch("true_beam_daughter_reco_byHits_PFP_nHits", &true_beam_daughter_reco_byHits_PFP_nHits);
fTree->Branch("true_beam_daughter_reco_byHits_PFP_trackScore", &true_beam_daughter_reco_byHits_PFP_trackScore);
fTree->Branch("true_beam_daughter_reco_byHits_allTrack_ID", &true_beam_daughter_reco_byHits_allTrack_ID);
fTree->Branch("true_beam_daughter_reco_byHits_allTrack_startX", &true_beam_daughter_reco_byHits_allTrack_startX);
fTree->Branch("true_beam_daughter_reco_byHits_allTrack_startY", &true_beam_daughter_reco_byHits_allTrack_startY);
fTree->Branch("true_beam_daughter_reco_byHits_allTrack_startZ", &true_beam_daughter_reco_byHits_allTrack_startZ);
fTree->Branch("true_beam_daughter_reco_byHits_allTrack_endX", &true_beam_daughter_reco_byHits_allTrack_endX);
fTree->Branch("true_beam_daughter_reco_byHits_allTrack_endY", &true_beam_daughter_reco_byHits_allTrack_endY);
fTree->Branch("true_beam_daughter_reco_byHits_allTrack_endZ", &true_beam_daughter_reco_byHits_allTrack_endZ);
fTree->Branch("true_beam_daughter_reco_byHits_allTrack_len", &true_beam_daughter_reco_byHits_allTrack_len);

fTree->Branch("true_beam_daughter_reco_byHits_allShower_ID", &true_beam_daughter_reco_byHits_allShower_ID);
fTree->Branch("true_beam_daughter_reco_byHits_allShower_startX", &true_beam_daughter_reco_byHits_allShower_startX);
fTree->Branch("true_beam_daughter_reco_byHits_allShower_startY", &true_beam_daughter_reco_byHits_allShower_startY);
fTree->Branch("true_beam_daughter_reco_byHits_allShower_startZ", &true_beam_daughter_reco_byHits_allShower_startZ);
fTree->Branch("true_beam_daughter_reco_byHits_allShower_len", &true_beam_daughter_reco_byHits_allShower_len);

fTree->Branch("true_beam_Pi0_decay_ID", &true_beam_Pi0_decay_ID);
fTree->Branch("true_beam_Pi0_decay_parID", &true_beam_Pi0_decay_parID);
fTree->Branch("true_beam_Pi0_decay_PDG", &true_beam_Pi0_decay_PDG);
fTree->Branch("true_beam_Pi0_decay_startP", &true_beam_Pi0_decay_startP);
fTree->Branch("true_beam_Pi0_decay_len", &true_beam_Pi0_decay_len);
fTree->Branch("true_beam_Pi0_decay_nHits", &true_beam_Pi0_decay_nHits);
fTree->Branch("true_beam_Pi0_decay_reco_byHits_PFP_ID", &true_beam_Pi0_decay_reco_byHits_PFP_ID);
fTree->Branch("true_beam_Pi0_decay_reco_byHits_PFP_nHits", &true_beam_Pi0_decay_reco_byHits_PFP_nHits);
fTree->Branch("true_beam_Pi0_decay_reco_byHits_PFP_trackScore", &true_beam_Pi0_decay_reco_byHits_PFP_trackScore);

fTree->Branch("true_beam_Pi0_decay_reco_byHits_allTrack_ID", &true_beam_Pi0_decay_reco_byHits_allTrack_ID);
fTree->Branch("true_beam_Pi0_decay_reco_byHits_allTrack_startX", &true_beam_Pi0_decay_reco_byHits_allTrack_startX);
fTree->Branch("true_beam_Pi0_decay_reco_byHits_allTrack_startY", &true_beam_Pi0_decay_reco_byHits_allTrack_startY);
fTree->Branch("true_beam_Pi0_decay_reco_byHits_allTrack_startZ", &true_beam_Pi0_decay_reco_byHits_allTrack_startZ);
fTree->Branch("true_beam_Pi0_decay_reco_byHits_allTrack_endX", &true_beam_Pi0_decay_reco_byHits_allTrack_endX);
fTree->Branch("true_beam_Pi0_decay_reco_byHits_allTrack_endY", &true_beam_Pi0_decay_reco_byHits_allTrack_endY);
fTree->Branch("true_beam_Pi0_decay_reco_byHits_allTrack_endZ", &true_beam_Pi0_decay_reco_byHits_allTrack_endZ);
fTree->Branch("true_beam_Pi0_decay_reco_byHits_allTrack_len", &true_beam_Pi0_decay_reco_byHits_allTrack_len);

fTree->Branch("true_beam_Pi0_decay_reco_byHits_allShower_ID", &true_beam_Pi0_decay_reco_byHits_allShower_ID);
fTree->Branch("true_beam_Pi0_decay_reco_byHits_allShower_startX", &true_beam_Pi0_decay_reco_byHits_allShower_startX);
fTree->Branch("true_beam_Pi0_decay_reco_byHits_allShower_startY", &true_beam_Pi0_decay_reco_byHits_allShower_startY);
fTree->Branch("true_beam_Pi0_decay_reco_byHits_allShower_startZ", &true_beam_Pi0_decay_reco_byHits_allShower_startZ);
fTree->Branch("true_beam_Pi0_decay_reco_byHits_allShower_len", &true_beam_Pi0_decay_reco_byHits_allShower_len);

fTree->Branch("true_beam_grand_daughter_ID", &true_beam_grand_daughter_ID);
fTree->Branch("true_beam_grand_daughter_parID", &true_beam_grand_daughter_parID);
fTree->Branch("true_beam_grand_daughter_PDG", &true_beam_grand_daughter_PDG);
fTree->Branch("true_beam_grand_daughter_nHits", &true_beam_grand_daughter_nHits);
fTree->Branch("true_beam_grand_daughter_Process", &true_beam_grand_daughter_Process);
fTree->Branch("true_beam_grand_daughter_endProcess", &true_beam_grand_daughter_endProcess);
```

reco daughter track

```
fTree->Branch("reco_daughter_trackID", &reco_daughter_trackID);
fTree->Branch("reco_daughter_true_byE_completeness", &reco_daughter_true_byE_completeness);
fTree->Branch("reco_daughter_true_byE_purity", &reco_daughter_true_byE_purity);
fTree->Branch("reco_daughter_true_byE_PDG", &reco_daughter_true_byE_PDG);
fTree->Branch("reco_daughter_true_byE_ID", &reco_daughter_true_byE_ID);
fTree->Branch("reco_daughter_true_byE_origin", &reco_daughter_true_byE_origin);
fTree->Branch("reco_daughter_true_byE_parID", &reco_daughter_true_byE_parID);
fTree->Branch("reco_daughter_true_byE_parPDG", &reco_daughter_true_byE_parPDG);
fTree->Branch("reco_daughter_true_byE_process", &reco_daughter_true_byE_process);

fTree->Branch("reco_daughter_true_byHits_PDG", &reco_daughter_true_byHits_PDG);
fTree->Branch("reco_daughter_true_byHits_ID", &reco_daughter_true_byHits_ID);
fTree->Branch("reco_daughter_true_byHits_origin", &reco_daughter_true_byHits_origin);
fTree->Branch("reco_daughter_true_byHits_parID", &reco_daughter_true_byHits_parID);
fTree->Branch("reco_daughter_true_byHits_parPDG", &reco_daughter_true_byHits_parPDG);
fTree->Branch("reco_daughter_true_byHits_process", &reco_daughter_true_byHits_process);
fTree->Branch("reco_daughter_true_byHits_purity", &reco_daughter_true_byHits_purity);
fTree->Branch("reco_daughter_true_byHits_sharedHits", &reco_daughter_true_byHits_sharedHits);
fTree->Branch("reco_daughter_true_byHits_emHits", &reco_daughter_true_byHits_emHits);

fTree->Branch("reco_daughter_true_byHits_len", &reco_daughter_true_byHits_len);
fTree->Branch("reco_daughter_true_byHits_startX", &reco_daughter_true_byHits_startX);
fTree->Branch("reco_daughter_true_byHits_startY", &reco_daughter_true_byHits_startY);
fTree->Branch("reco_daughter_true_byHits_startZ", &reco_daughter_true_byHits_startZ);
fTree->Branch("reco_daughter_true_byHits_endX", &reco_daughter_true_byHits_endX);
fTree->Branch("reco_daughter_true_byHits_endY", &reco_daughter_true_byHits_endY);
fTree->Branch("reco_daughter_true_byHits_endZ", &reco_daughter_true_byHits_endZ);

fTree->Branch("reco_daughter_true_byHits_startPx", &reco_daughter_true_byHits_startPx);
fTree->Branch("reco_daughter_true_byHits_startPy", &reco_daughter_true_byHits_startPy);
fTree->Branch("reco_daughter_true_byHits_startPz", &reco_daughter_true_byHits_startPz);
fTree->Branch("reco_daughter_true_byHits_startP", &reco_daughter_true_byHits_startP);
fTree->Branch("reco_daughter_true_byHits_startE", &reco_daughter_true_byHits_startE);
//Alternative reco
fTree->Branch("reco_daughter_PFP_true_byHits_PDG", &reco_daughter_PFP_true_byHits_PDG);
fTree->Branch("reco_daughter_PFP_true_byHits_ID", &reco_daughter_PFP_true_byHits_ID);
fTree->Branch("reco_daughter_PFP_true_byHits_origin", &reco_daughter_PFP_true_byHits_origin);
fTree->Branch("reco_daughter_PFP_true_byHits_parID", &reco_daughter_PFP_true_byHits_parID);
fTree->Branch("reco_daughter_PFP_true_byHits_parPDG", &reco_daughter_PFP_true_byHits_parPDG);
fTree->Branch("reco_daughter_PFP_true_byHits_process", &reco_daughter_PFP_true_byHits_process);
fTree->Branch("reco_daughter_PFP_true_byHits_sharedHits",
&reco_daughter_PFP_true_byHits_sharedHits);
fTree->Branch("reco_daughter_PFP_true_byHits_emHits", &reco_daughter_PFP_true_byHits_emHits);

fTree->Branch("reco_daughter_PFP_true_byHits_len", &reco_daughter_PFP_true_byHits_len);
fTree->Branch("reco_daughter_PFP_true_byHits_startX", &reco_daughter_PFP_true_byHits_startX);
fTree->Branch("reco_daughter_PFP_true_byHits_startY", &reco_daughter_PFP_true_byHits_startY);
fTree->Branch("reco_daughter_PFP_true_byHits_startZ", &reco_daughter_PFP_true_byHits_startZ);
fTree->Branch("reco_daughter_PFP_true_byHits_endX", &reco_daughter_PFP_true_byHits_endX);
fTree->Branch("reco_daughter_PFP_true_byHits_endY", &reco_daughter_PFP_true_byHits_endY);
fTree->Branch("reco_daughter_PFP_true_byHits_endZ", &reco_daughter_PFP_true_byHits_endZ);

fTree->Branch("reco_daughter_PFP_true_byHits_startPx", &reco_daughter_PFP_true_byHits_startPx);
fTree->Branch("reco_daughter_PFP_true_byHits_startPy", &reco_daughter_PFP_true_byHits_startPy);
fTree->Branch("reco_daughter_PFP_true_byHits_startPz", &reco_daughter_PFP_true_byHits_startPz);
fTree->Branch("reco_daughter_PFP_true_byHits_startP", &reco_daughter_PFP_true_byHits_startP);
fTree->Branch("reco_daughter_PFP_true_byHits_startE", &reco_daughter_PFP_true_byHits_startE);
fTree->Branch("reco_daughter_PFP_true_byHits_endProcess",
&reco_daughter_PFP_true_byHits_endProcess);
fTree->Branch("reco_daughter_PFP_true_byHits_purity", &reco_daughter_PFP_true_byHits_purity);

fTree->Branch("reco_daughter_allTrack_ID", &reco_daughter_allTrack_ID);
fTree->Branch("reco_daughter_allTrack_dEdX", &reco_daughter_allTrack_dEdX);
fTree->Branch("reco_daughter_allTrack_dQdX", &reco_daughter_allTrack_dQdX);
fTree->Branch("reco_daughter_allTrack_resRange", &reco_daughter_allTrack_resRange);
fTree->Branch("reco_daughter_allTrack_dQdX_SCE", &reco_daughter_allTrack_dQdX_SCE);
fTree->Branch("reco_daughter_allTrack_dEdX_SCE", &reco_daughter_allTrack_dEdX);
fTree->Branch("reco_daughter_allTrack_resRange_SCE", &reco_daughter_allTrack_resRange);

fTree->Branch("reco_daughter_allTrack_calibrated_dEdX", &reco_daughter_allTrack_calibrated_dEdX);
fTree->Branch("reco_daughter_allTrack_calibrated_dEdX_SCE",
&reco_daughter_allTrack_calibrated_dEdX_SCE);

fTree->Branch("reco_daughter_allTrack_Chi2_proton", &reco_daughter_allTrack_Chi2_proton);
fTree->Branch("reco_daughter_allTrack_Chi2_ndof", &reco_daughter_allTrack_Chi2_ndof);

fTree->Branch("reco_daughter_allTrack_Theta", &reco_daughter_allTrack_Theta);
fTree->Branch("reco_daughter_allTrack_Phi", &reco_daughter_allTrack_Phi);

fTree->Branch("reco_daughter_allTrack_len", &reco_daughter_allTrack_len);
```

```
fTree->Branch("reco_daughter_allTrack_startX", &reco_daughter_allTrack_startX);
fTree->Branch("reco_daughter_allTrack_startY", &reco_daughter_allTrack_startY);
fTree->Branch("reco_daughter_allTrack_startZ", &reco_daughter_allTrack_startZ);
fTree->Branch("reco_daughter_allTrack_endX", &reco_daughter_allTrack_endX);
fTree->Branch("reco_daughter_allTrack_endY", &reco_daughter_allTrack_endY);
fTree->Branch("reco_daughter_allTrack_endZ", &reco_daughter_allTrack_endZ);
fTree->Branch("reco_daughter_allTrack_dR", &reco_daughter_allTrack_dR);
fTree->Branch("reco_daughter_allTrack_to_vertex", &reco_daughter_allTrack_to_vertex);
//////

fTree->Branch("reco_daughter_allShower_ID", &reco_daughter_allShower_ID);
fTree->Branch("reco_daughter_allShower_len", &reco_daughter_allShower_len);
fTree->Branch("reco_daughter_allShower_startX", &reco_daughter_allShower_startX);
fTree->Branch("reco_daughter_allShower_startY", &reco_daughter_allShower_startY);
fTree->Branch("reco_daughter_allShower_startZ", &reco_daughter_allShower_startZ);


fTree->Branch("reco_daughter_shower_true_byE_PDG", &reco_daughter_shower_true_byE_PDG);
fTree->Branch("reco_daughter_shower_true_byE_ID", &reco_daughter_shower_true_byE_ID);
fTree->Branch("reco_daughter_shower_true_byE_origin", &reco_daughter_shower_true_byE_origin);
fTree->Branch("reco_daughter_shower_true_byE_parID", &reco_daughter_shower_true_byE_parID);
fTree->Branch("reco_daughter_shower_true_byE_parPDG", &reco_daughter_shower_true_byE_parPDG);

fTree->Branch("reco_daughter_shower_true_byE_startPx", &reco_daughter_shower_true_byE_startPx);
fTree->Branch("reco_daughter_shower_true_byE_startPy", &reco_daughter_shower_true_byE_startPy);
fTree->Branch("reco_daughter_shower_true_byE_startPz", &reco_daughter_shower_true_byE_startPz);
fTree->Branch("reco_daughter_shower_true_byE_startP", &reco_daughter_shower_true_byE_startP);
fTree->Branch("reco_daughter_shower_true_byE_endProcess", &reco_daughter_shower_true_byE_endProcess);


fTree->Branch("reco_daughter_shower_true_byHits_PDG", &reco_daughter_shower_true_byHits_PDG);
fTree->Branch("reco_daughter_shower_true_byHits_ID", &reco_daughter_shower_true_byHits_ID);
fTree->Branch("reco_daughter_shower_true_byHits_origin", &reco_daughter_shower_true_byHits_origin);
fTree->Branch("reco_daughter_shower_true_byHits_parID", &reco_daughter_shower_true_byHits_parID);
fTree->Branch("reco_daughter_shower_true_byHits_parPDG", &reco_daughter_shower_true_byHits_parPDG);
fTree->Branch("reco_daughter_shower_true_byHits_process", &reco_daughter_shower_true_byHits_process);
fTree->Branch("reco_daughter_shower_true_byHits_purity", &reco_daughter_shower_true_byHits_purity);

fTree->Branch("reco_daughter_shower_true_byHits_startPx", &reco_daughter_shower_true_byHits_startPx);
fTree->Branch("reco_daughter_shower_true_byHits_startPy", &reco_daughter_shower_true_byHits_startPy);
fTree->Branch("reco_daughter_shower_true_byHits_startPz", &reco_daughter_shower_true_byHits_startPz);
fTree->Branch("reco_daughter_shower_true_byHits_startP", &reco_daughter_shower_true_byHits_startP);
fTree->Branch("reco_daughter_shower_true_byHits_endProcess", &reco_daughter_shower_true_byHits_endProcess);


///Reconstructed info -- daughter
fTree->Branch("reco_daughter_showerID", &reco_daughter_showerID);
fTree->Branch("reco_daughter_dQdX", &reco_daughter_dQdX);
fTree->Branch("reco_daughter_dEdX", &reco_daughter_dEdX);
fTree->Branch("reco_daughter_resRange", &reco_daughter_resRange);
fTree->Branch("reco_daughter_shower_dQdX", &reco_daughter_shower_dQdX);
fTree->Branch("reco_daughter_shower_dEdX", &reco_daughter_shower_dEdX);
fTree->Branch("reco_daughter_shower_resRange", &reco_daughter_shower_resRange);
fTree->Branch("reco_daughter_len", &reco_daughter_len);
fTree->Branch("reco_daughter_startX", &reco_daughter_startX);
fTree->Branch("reco_daughter_startY", &reco_daughter_startY);
fTree->Branch("reco_daughter_startZ", &reco_daughter_startZ);
fTree->Branch("reco_daughter_endX", &reco_daughter_endX);
fTree->Branch("reco_daughter_endY", &reco_daughter_endY);
fTree->Branch("reco_daughter_endZ", &reco_daughter_endZ);
fTree->Branch("reco_daughter_deltaR", &reco_daughter_deltaR);

fTree->Branch("reco_daughter_dR", &reco_daughter_dR);
fTree->Branch("reco_daughter_to_vertex", &reco_daughter_to_vertex);
fTree->Branch("reco_daughter_slice", &reco_daughter_slice);

fTree->Branch("reco_daughter_shower_to_vertex", &reco_daughter_shower_to_vertex);

fTree->Branch("reco_daughter_shower_startX", &reco_daughter_shower_startX);
fTree->Branch("reco_daughter_shower_startY", &reco_daughter_shower_startY);
fTree->Branch("reco_daughter_shower_startZ", &reco_daughter_shower_startZ);
fTree->Branch("reco_daughter_shower_len", &reco_daughter_shower_len);


fTree->Branch("reco_daughter_PFP_ID", &reco_daughter_PFP_ID);
fTree->Branch("reco_daughter_PFP_nHits", &reco_daughter_PFP_nHits);
fTree->Branch("reco_daughter_PFP_trackScore", &reco_daughter_PFP_trackScore);
fTree->Branch("reco_daughter_PFP_emScore", &reco_daughter_PFP_emScore);
fTree->Branch("reco_daughter_PFP_michelScore", &reco_daughter_PFP_michelScore);
```

reco daughter track

```
fTree->Branch("reco_daughter_trackID", &reco_daughter_trackID);
fTree->Branch("reco_daughter_true_byE_completeness", &reco_daughter_true_byE_completeness);
fTree->Branch("reco_daughter_true_byE_purity", &reco_daughter_true_byE_purity);
fTree->Branch("reco_daughter_true_byE_PDG", &reco_daughter_true_byE_PDG);
fTree->Branch("reco_daughter_true_byE_ID", &reco_daughter_true_byE_ID);
fTree->Branch("reco_daughter_true_byE_origin", &reco_daughter_true_byE_origin);
fTree->Branch("reco_daughter_true_byE_parID", &reco_daughter_true_byE_parID);
fTree->Branch("reco_daughter_true_byE_parPDG", &reco_daughter_true_byE_parPDG);
fTree->Branch("reco_daughter_true_byE_process", &reco_daughter_true_byE_process);

fTree->Branch("reco_daughter_true_byHits_PDG", &reco_daughter_true_byHits_PDG);
fTree->Branch("reco_daughter_true_byHits_ID", &reco_daughter_true_byHits_ID);
fTree->Branch("reco_daughter_true_byHits_origin", &reco_daughter_true_byHits_origin);
fTree->Branch("reco_daughter_true_byHits_parID", &reco_daughter_true_byHits_parID);
fTree->Branch("reco_daughter_true_byHits_parPDG", &reco_daughter_true_byHits_parPDG);
fTree->Branch("reco_daughter_true_byHits_process", &reco_daughter_true_byHits_process);
fTree->Branch("reco_daughter_true_byHits_purity", &reco_daughter_true_byHits_purity);
fTree->Branch("reco_daughter_true_byHits_sharedHits", &reco_daughter_true_byHits_sharedHits);
fTree->Branch("reco_daughter_true_byHits_emHits", &reco_daughter_true_byHits_emHits);

fTree->Branch("reco_daughter_true_byHits_len", &reco_daughter_true_byHits_len);
fTree->Branch("reco_daughter_true_byHits_startX", &reco_daughter_true_byHits_startX);
fTree->Branch("reco_daughter_true_byHits_startY", &reco_daughter_true_byHits_startY);
fTree->Branch("reco_daughter_true_byHits_startZ", &reco_daughter_true_byHits_startZ);
fTree->Branch("reco_daughter_true_byHits_endX", &reco_daughter_true_byHits_endX);
fTree->Branch("reco_daughter_true_byHits_endY", &reco_daughter_true_byHits_endY);
fTree->Branch("reco_daughter_true_byHits_endZ", &reco_daughter_true_byHits_endZ);

fTree->Branch("reco_daughter_true_byHits_startPx", &reco_daughter_true_byHits_startPx);
fTree->Branch("reco_daughter_true_byHits_startPy", &reco_daughter_true_byHits_startPy);
fTree->Branch("reco_daughter_true_byHits_startPz", &reco_daughter_true_byHits_startPz);
fTree->Branch("reco_daughter_true_byHits_startP", &reco_daughter_true_byHits_startP);
fTree->Branch("reco_daughter_true_byHits_startE", &reco_daughter_true_byHits_startE);
//Alternative reco
fTree->Branch("reco_daughter_PFP_true_byHits_PDG", &reco_daughter_PFP_true_byHits_PDG);
fTree->Branch("reco_daughter_PFP_true_byHits_ID", &reco_daughter_PFP_true_byHits_ID);
fTree->Branch("reco_daughter_PFP_true_byHits_origin", &reco_daughter_PFP_true_byHits_origin);
fTree->Branch("reco_daughter_PFP_true_byHits_parID", &reco_daughter_PFP_true_byHits_parID);
fTree->Branch("reco_daughter_PFP_true_byHits_parPDG", &reco_daughter_PFP_true_byHits_parPDG);
fTree->Branch("reco_daughter_PFP_true_byHits_process", &reco_daughter_PFP_true_byHits_process);
fTree->Branch("reco_daughter_PFP_true_byHits_sharedHits",
&reco_daughter_PFP_true_byHits_sharedHits);
fTree->Branch("reco_daughter_PFP_true_byHits_emHits", &reco_daughter_PFP_true_byHits_emHits);

fTree->Branch("reco_daughter_PFP_true_byHits_len", &reco_daughter_PFP_true_byHits_len);
fTree->Branch("reco_daughter_PFP_true_byHits_startX", &reco_daughter_PFP_true_byHits_startX);
fTree->Branch("reco_daughter_PFP_true_byHits_startY", &reco_daughter_PFP_true_byHits_startY);
fTree->Branch("reco_daughter_PFP_true_byHits_startZ", &reco_daughter_PFP_true_byHits_startZ);
fTree->Branch("reco_daughter_PFP_true_byHits_endX", &reco_daughter_PFP_true_byHits_endX);
fTree->Branch("reco_daughter_PFP_true_byHits_endY", &reco_daughter_PFP_true_byHits_endY);
fTree->Branch("reco_daughter_PFP_true_byHits_endZ", &reco_daughter_PFP_true_byHits_endZ);

fTree->Branch("reco_daughter_PFP_true_byHits_startPx", &reco_daughter_PFP_true_byHits_startPx);
fTree->Branch("reco_daughter_PFP_true_byHits_startPy", &reco_daughter_PFP_true_byHits_startPy);
fTree->Branch("reco_daughter_PFP_true_byHits_startPz", &reco_daughter_PFP_true_byHits_startPz);
fTree->Branch("reco_daughter_PFP_true_byHits_startP", &reco_daughter_PFP_true_byHits_startP);
fTree->Branch("reco_daughter_PFP_true_byHits_startE", &reco_daughter_PFP_true_byHits_startE);
fTree->Branch("reco_daughter_PFP_true_byHits_endProcess",
&reco_daughter_PFP_true_byHits_endProcess);
fTree->Branch("reco_daughter_PFP_true_byHits_purity", &reco_daughter_PFP_true_byHits_purity);

fTree->Branch("reco_daughter_allTrack_ID", &reco_daughter_allTrack_ID);
fTree->Branch("reco_daughter_allTrack_dEdX", &reco_daughter_allTrack_dEdX);
fTree->Branch("reco_daughter_allTrack_dQdX", &reco_daughter_allTrack_dQdX);
fTree->Branch("reco_daughter_allTrack_resRange", &reco_daughter_allTrack_resRange);
fTree->Branch("reco_daughter_allTrack_dQdX_SCE", &reco_daughter_allTrack_dQdX_SCE);
fTree->Branch("reco_daughter_allTrack_dEdX_SCE", &reco_daughter_allTrack_dEdX);
fTree->Branch("reco_daughter_allTrack_resRange_SCE", &reco_daughter_allTrack_resRange);

fTree->Branch("reco_daughter_allTrack_calibrated_dEdX", &reco_daughter_allTrack_calibrated_dEdX);
fTree->Branch("reco_daughter_allTrack_calibrated_dEdX_SCE",
&reco_daughter_allTrack_calibrated_dEdX_SCE);

fTree->Branch("reco_daughter_allTrack_Chi2_proton", &reco_daughter_allTrack_Chi2_proton);
fTree->Branch("reco_daughter_allTrack_Chi2_ndof", &reco_daughter_allTrack_Chi2_ndof);

fTree->Branch("reco_daughter_allTrack_Theta", &reco_daughter_allTrack_Theta);
fTree->Branch("reco_daughter_allTrack_Phi", &reco_daughter_allTrack_Phi);

fTree->Branch("reco_daughter_allTrack_len", &reco_daughter_allTrack_len);
```

```
fTree->Branch("reco_daughter_allTrack_startX", &reco_daughter_allTrack_startX);
fTree->Branch("reco_daughter_allTrack_startY", &reco_daughter_allTrack_startY);
fTree->Branch("reco_daughter_allTrack_startZ", &reco_daughter_allTrack_startZ);
fTree->Branch("reco_daughter_allTrack_endX", &reco_daughter_allTrack_endX);
fTree->Branch("reco_daughter_allTrack_endY", &reco_daughter_allTrack_endY);
fTree->Branch("reco_daughter_allTrack_endZ", &reco_daughter_allTrack_endZ);
fTree->Branch("reco_daughter_allTrack_dR", &reco_daughter_allTrack_dR);
fTree->Branch("reco_daughter_allTrack_to_vertex", &reco_daughter_allTrack_to_vertex);
//////

fTree->Branch("reco_daughter_allShower_ID", &reco_daughter_allShower_ID);
fTree->Branch("reco_daughter_allShower_len", &reco_daughter_allShower_len);
fTree->Branch("reco_daughter_allShower_startX", &reco_daughter_allShower_startX);
fTree->Branch("reco_daughter_allShower_startY", &reco_daughter_allShower_startY);
fTree->Branch("reco_daughter_allShower_startZ", &reco_daughter_allShower_startZ);


fTree->Branch("reco_daughter_shower_true_byE_PDG", &reco_daughter_shower_true_byE_PDG);
fTree->Branch("reco_daughter_shower_true_byE_ID", &reco_daughter_shower_true_byE_ID);
fTree->Branch("reco_daughter_shower_true_byE_origin", &reco_daughter_shower_true_byE_origin);
fTree->Branch("reco_daughter_shower_true_byE_parID", &reco_daughter_shower_true_byE_parID);
fTree->Branch("reco_daughter_shower_true_byE_parPDG", &reco_daughter_shower_true_byE_parPDG);

fTree->Branch("reco_daughter_shower_true_byE_startPx", &reco_daughter_shower_true_byE_startPx);
fTree->Branch("reco_daughter_shower_true_byE_startPy", &reco_daughter_shower_true_byE_startPy);
fTree->Branch("reco_daughter_shower_true_byE_startPz", &reco_daughter_shower_true_byE_startPz);
fTree->Branch("reco_daughter_shower_true_byE_startP", &reco_daughter_shower_true_byE_startP);
fTree->Branch("reco_daughter_shower_true_byE_endProcess", &reco_daughter_shower_true_byE_endProcess);


fTree->Branch("reco_daughter_shower_true_byHits_PDG", &reco_daughter_shower_true_byHits_PDG);
fTree->Branch("reco_daughter_shower_true_byHits_ID", &reco_daughter_shower_true_byHits_ID);
fTree->Branch("reco_daughter_shower_true_byHits_origin", &reco_daughter_shower_true_byHits_origin);
fTree->Branch("reco_daughter_shower_true_byHits_parID", &reco_daughter_shower_true_byHits_parID);
fTree->Branch("reco_daughter_shower_true_byHits_parPDG", &reco_daughter_shower_true_byHits_parPDG);
fTree->Branch("reco_daughter_shower_true_byHits_process", &reco_daughter_shower_true_byHits_process);
fTree->Branch("reco_daughter_shower_true_byHits_purity", &reco_daughter_shower_true_byHits_purity);

fTree->Branch("reco_daughter_shower_true_byHits_startPx", &reco_daughter_shower_true_byHits_startPx);
fTree->Branch("reco_daughter_shower_true_byHits_startPy", &reco_daughter_shower_true_byHits_startPy);
fTree->Branch("reco_daughter_shower_true_byHits_startPz", &reco_daughter_shower_true_byHits_startPz);
fTree->Branch("reco_daughter_shower_true_byHits_startP", &reco_daughter_shower_true_byHits_startP);
fTree->Branch("reco_daughter_shower_true_byHits_endProcess", &reco_daughter_shower_true_byHits_endProcess);


///Reconstructed info -- daughter
fTree->Branch("reco_daughter_showerID", &reco_daughter_showerID);
fTree->Branch("reco_daughter_dQdX", &reco_daughter_dQdX);
fTree->Branch("reco_daughter_dEdX", &reco_daughter_dEdX);
fTree->Branch("reco_daughter_resRange", &reco_daughter_resRange);
fTree->Branch("reco_daughter_shower_dQdX", &reco_daughter_shower_dQdX);
fTree->Branch("reco_daughter_shower_dEdX", &reco_daughter_shower_dEdX);
fTree->Branch("reco_daughter_shower_resRange", &reco_daughter_shower_resRange);
fTree->Branch("reco_daughter_len", &reco_daughter_len);
fTree->Branch("reco_daughter_startX", &reco_daughter_startX);
fTree->Branch("reco_daughter_startY", &reco_daughter_startY);
fTree->Branch("reco_daughter_startZ", &reco_daughter_startZ);
fTree->Branch("reco_daughter_endX", &reco_daughter_endX);
fTree->Branch("reco_daughter_endY", &reco_daughter_endY);
fTree->Branch("reco_daughter_endZ", &reco_daughter_endZ);
fTree->Branch("reco_daughter_deltaR", &reco_daughter_deltaR);

fTree->Branch("reco_daughter_dR", &reco_daughter_dR);
fTree->Branch("reco_daughter_to_vertex", &reco_daughter_to_vertex);
fTree->Branch("reco_daughter_slice", &reco_daughter_slice);

fTree->Branch("reco_daughter_shower_to_vertex", &reco_daughter_shower_to_vertex);

fTree->Branch("reco_daughter_shower_startX", &reco_daughter_shower_startX);
fTree->Branch("reco_daughter_shower_startY", &reco_daughter_shower_startY);
fTree->Branch("reco_daughter_shower_startZ", &reco_daughter_shower_startZ);
fTree->Branch("reco_daughter_shower_len", &reco_daughter_shower_len);


fTree->Branch("reco_daughter_PFP_ID", &reco_daughter_PFP_ID);
fTree->Branch("reco_daughter_PFP_nHits", &reco_daughter_PFP_nHits);
fTree->Branch("reco_daughter_PFP_trackScore", &reco_daughter_PFP_trackScore);
fTree->Branch("reco_daughter_PFP_emScore", &reco_daughter_PFP_emScore);
fTree->Branch("reco_daughter_PFP_michelScore", &reco_daughter_PFP_michelScore);
```

reco daughter track

```
fTree->Branch("reco_daughter_trackID", &reco_daughter_trackID);
fTree->Branch("reco_daughter_true_byE_completeness", &reco_daughter_true_byE_completeness);
fTree->Branch("reco_daughter_true_byE_purity", &reco_daughter_true_byE_purity);
fTree->Branch("reco_daughter_true_byE_PDG", &reco_daughter_true_byE_PDG);
fTree->Branch("reco_daughter_true_byE_ID", &reco_daughter_true_byE_ID);
fTree->Branch("reco_daughter_true_byE_origin", &reco_daughter_true_byE_origin);
fTree->Branch("reco_daughter_true_byE_parID", &reco_daughter_true_byE_parID);
fTree->Branch("reco_daughter_true_byE_parPDG", &reco_daughter_true_byE_parPDG);
fTree->Branch("reco_daughter_true_byE_process", &reco_daughter_true_byE_process);

fTree->Branch("reco_daughter_true_byHits_PDG", &reco_daughter_true_byHits_PDG);
fTree->Branch("reco_daughter_true_byHits_ID", &reco_daughter_true_byHits_ID);
fTree->Branch("reco_daughter_true_byHits_origin", &reco_daughter_true_byHits_origin);
fTree->Branch("reco_daughter_true_byHits_parID", &reco_daughter_true_byHits_parID);
fTree->Branch("reco_daughter_true_byHits_parPDG", &reco_daughter_true_byHits_parPDG);
fTree->Branch("reco_daughter_true_byHits_process", &reco_daughter_true_byHits_process);
fTree->Branch("reco_daughter_true_byHits_purity", &reco_daughter_true_byHits_purity);
fTree->Branch("reco_daughter_true_byHits_sharedHits", &reco_daughter_true_byHits_sharedHits);
fTree->Branch("reco_daughter_true_byHits_emHits", &reco_daughter_true_byHits_emHits);

fTree->Branch("reco_daughter_true_byHits_len", &reco_daughter_true_byHits_len);
fTree->Branch("reco_daughter_true_byHits_startX", &reco_daughter_true_byHits_startX);
fTree->Branch("reco_daughter_true_byHits_startY", &reco_daughter_true_byHits_startY);
fTree->Branch("reco_daughter_true_byHits_startZ", &reco_daughter_true_byHits_startZ);
fTree->Branch("reco_daughter_true_byHits_endX", &reco_daughter_true_byHits_endX);
fTree->Branch("reco_daughter_true_byHits_endY", &reco_daughter_true_byHits_endY);
fTree->Branch("reco_daughter_true_byHits_endZ", &reco_daughter_true_byHits_endZ);

fTree->Branch("reco_daughter_true_byHits_startPx", &reco_daughter_true_byHits_startPx);
fTree->Branch("reco_daughter_true_byHits_startPy", &reco_daughter_true_byHits_startPy);
fTree->Branch("reco_daughter_true_byHits_startPz", &reco_daughter_true_byHits_startPz);
fTree->Branch("reco_daughter_true_byHits_startP", &reco_daughter_true_byHits_startP);
fTree->Branch("reco_daughter_true_byHits_startE", &reco_daughter_true_byHits_startE);
//Alternative reco
fTree->Branch("reco_daughter_PFP_true_byHits_PDG", &reco_daughter_PFP_true_byHits_PDG);
fTree->Branch("reco_daughter_PFP_true_byHits_ID", &reco_daughter_PFP_true_byHits_ID);
fTree->Branch("reco_daughter_PFP_true_byHits_origin", &reco_daughter_PFP_true_byHits_origin);
fTree->Branch("reco_daughter_PFP_true_byHits_parID", &reco_daughter_PFP_true_byHits_parID);
fTree->Branch("reco_daughter_PFP_true_byHits_parPDG", &reco_daughter_PFP_true_byHits_parPDG);
fTree->Branch("reco_daughter_PFP_true_byHits_process", &reco_daughter_PFP_true_byHits_process);
fTree->Branch("reco_daughter_PFP_true_byHits_sharedHits",
&reco_daughter_PFP_true_byHits_sharedHits);
fTree->Branch("reco_daughter_PFP_true_byHits_emHits", &reco_daughter_PFP_true_byHits_emHits);

fTree->Branch("reco_daughter_PFP_true_byHits_len", &reco_daughter_PFP_true_byHits_len);
fTree->Branch("reco_daughter_PFP_true_byHits_startX", &reco_daughter_PFP_true_byHits_startX);
fTree->Branch("reco_daughter_PFP_true_byHits_startY", &reco_daughter_PFP_true_byHits_startY);
fTree->Branch("reco_daughter_PFP_true_byHits_startZ", &reco_daughter_PFP_true_byHits_startZ);
fTree->Branch("reco_daughter_PFP_true_byHits_endX", &reco_daughter_PFP_true_byHits_endX);
fTree->Branch("reco_daughter_PFP_true_byHits_endY", &reco_daughter_PFP_true_byHits_endY);
fTree->Branch("reco_daughter_PFP_true_byHits_endZ", &reco_daughter_PFP_true_byHits_endZ);

fTree->Branch("reco_daughter_PFP_true_byHits_startPx", &reco_daughter_PFP_true_byHits_startPx);
fTree->Branch("reco_daughter_PFP_true_byHits_startPy", &reco_daughter_PFP_true_byHits_startPy);
fTree->Branch("reco_daughter_PFP_true_byHits_startPz", &reco_daughter_PFP_true_byHits_startPz);
fTree->Branch("reco_daughter_PFP_true_byHits_startP", &reco_daughter_PFP_true_byHits_startP);
fTree->Branch("reco_daughter_PFP_true_byHits_startE", &reco_daughter_PFP_true_byHits_startE);
fTree->Branch("reco_daughter_PFP_true_byHits_endProcess",
&reco_daughter_PFP_true_byHits_endProcess);
fTree->Branch("reco_daughter_PFP_true_byHits_purity", &reco_daughter_PFP_true_byHits_purity);

fTree->Branch("reco_daughter_allTrack_ID", &reco_daughter_allTrack_ID);
fTree->Branch("reco_daughter_allTrack_dEdX", &reco_daughter_allTrack_dEdX);
fTree->Branch("reco_daughter_allTrack_dQdX", &reco_daughter_allTrack_dQdX);
fTree->Branch("reco_daughter_allTrack_resRange", &reco_daughter_allTrack_resRange);
fTree->Branch("reco_daughter_allTrack_dQdX_SCE", &reco_daughter_allTrack_dQdX_SCE);
fTree->Branch("reco_daughter_allTrack_dEdX_SCE", &reco_daughter_allTrack_dEdX);
fTree->Branch("reco_daughter_allTrack_resRange_SCE", &reco_daughter_allTrack_resRange);

fTree->Branch("reco_daughter_allTrack_calibrated_dEdX", &reco_daughter_allTrack_calibrated_dEdX);
fTree->Branch("reco_daughter_allTrack_calibrated_dEdX_SCE",
&reco_daughter_allTrack_calibrated_dEdX_SCE);

fTree->Branch("reco_daughter_allTrack_Chi2_proton", &reco_daughter_allTrack_Chi2_proton);
fTree->Branch("reco_daughter_allTrack_Chi2_ndof", &reco_daughter_allTrack_Chi2_ndof);

fTree->Branch("reco_daughter_allTrack_Theta", &reco_daughter_allTrack_Theta);
fTree->Branch("reco_daughter_allTrack_Phi", &reco_daughter_allTrack_Phi);

fTree->Branch("reco_daughter_allTrack_len", &reco_daughter_allTrack_len);
```

```
fTree->Branch("reco_daughter_allTrack_startX", &reco_daughter_allTrack_startX);
fTree->Branch("reco_daughter_allTrack_startY", &reco_daughter_allTrack_startY);
fTree->Branch("reco_daughter_allTrack_startZ", &reco_daughter_allTrack_startZ);
fTree->Branch("reco_daughter_allTrack_endX", &reco_daughter_allTrack_endX);
fTree->Branch("reco_daughter_allTrack_endY", &reco_daughter_allTrack_endY);
fTree->Branch("reco_daughter_allTrack_endZ", &reco_daughter_allTrack_endZ);
fTree->Branch("reco_daughter_allTrack_dR", &reco_daughter_allTrack_dR);
fTree->Branch("reco_daughter_allTrack_to_vertex", &reco_daughter_allTrack_to_vertex);
//////

fTree->Branch("reco_daughter_allShower_ID", &reco_daughter_allShower_ID);
fTree->Branch("reco_daughter_allShower_len", &reco_daughter_allShower_len);
fTree->Branch("reco_daughter_allShower_startX", &reco_daughter_allShower_startX);
fTree->Branch("reco_daughter_allShower_startY", &reco_daughter_allShower_startY);
fTree->Branch("reco_daughter_allShower_startZ", &reco_daughter_allShower_startZ);


fTree->Branch("reco_daughter_shower_true_byE_PDG", &reco_daughter_shower_true_byE_PDG);
fTree->Branch("reco_daughter_shower_true_byE_ID", &reco_daughter_shower_true_byE_ID);
fTree->Branch("reco_daughter_shower_true_byE_origin", &reco_daughter_shower_true_byE_origin);
fTree->Branch("reco_daughter_shower_true_byE_parID", &reco_daughter_shower_true_byE_parID);
fTree->Branch("reco_daughter_shower_true_byE_parPDG", &reco_daughter_shower_true_byE_parPDG);

fTree->Branch("reco_daughter_shower_true_byE_startPx", &reco_daughter_shower_true_byE_startPx);
fTree->Branch("reco_daughter_shower_true_byE_startPy", &reco_daughter_shower_true_byE_startPy);
fTree->Branch("reco_daughter_shower_true_byE_startPz", &reco_daughter_shower_true_byE_startPz);
fTree->Branch("reco_daughter_shower_true_byE_startP", &reco_daughter_shower_true_byE_startP);
fTree->Branch("reco_daughter_shower_true_byE_endProcess", &reco_daughter_shower_true_byE_endProcess);


fTree->Branch("reco_daughter_shower_true_byHits_PDG", &reco_daughter_shower_true_byHits_PDG);
fTree->Branch("reco_daughter_shower_true_byHits_ID", &reco_daughter_shower_true_byHits_ID);
fTree->Branch("reco_daughter_shower_true_byHits_origin", &reco_daughter_shower_true_byHits_origin);
fTree->Branch("reco_daughter_shower_true_byHits_parID", &reco_daughter_shower_true_byHits_parID);
fTree->Branch("reco_daughter_shower_true_byHits_parPDG", &reco_daughter_shower_true_byHits_parPDG);
fTree->Branch("reco_daughter_shower_true_byHits_process", &reco_daughter_shower_true_byHits_process);
fTree->Branch("reco_daughter_shower_true_byHits_purity", &reco_daughter_shower_true_byHits_purity);

fTree->Branch("reco_daughter_shower_true_byHits_startPx", &reco_daughter_shower_true_byHits_startPx);
fTree->Branch("reco_daughter_shower_true_byHits_startPy", &reco_daughter_shower_true_byHits_startPy);
fTree->Branch("reco_daughter_shower_true_byHits_startPz", &reco_daughter_shower_true_byHits_startPz);
fTree->Branch("reco_daughter_shower_true_byHits_startP", &reco_daughter_shower_true_byHits_startP);
fTree->Branch("reco_daughter_shower_true_byHits_endProcess", &reco_daughter_shower_true_byHits_endProcess);


///Reconstructed info -- daughter
fTree->Branch("reco_daughter_showerID", &reco_daughter_showerID);
fTree->Branch("reco_daughter_dQdX", &reco_daughter_dQdX);
fTree->Branch("reco_daughter_dEdX", &reco_daughter_dEdX);
fTree->Branch("reco_daughter_resRange", &reco_daughter_resRange);
fTree->Branch("reco_daughter_shower_dQdX", &reco_daughter_shower_dQdX);
fTree->Branch("reco_daughter_shower_dEdX", &reco_daughter_shower_dEdX);
fTree->Branch("reco_daughter_shower_resRange", &reco_daughter_shower_resRange);
fTree->Branch("reco_daughter_len", &reco_daughter_len);
fTree->Branch("reco_daughter_startX", &reco_daughter_startX);
fTree->Branch("reco_daughter_startY", &reco_daughter_startY);
fTree->Branch("reco_daughter_startZ", &reco_daughter_startZ);
fTree->Branch("reco_daughter_endX", &reco_daughter_endX);
fTree->Branch("reco_daughter_endY", &reco_daughter_endY);
fTree->Branch("reco_daughter_endZ", &reco_daughter_endZ);
fTree->Branch("reco_daughter_deltaR", &reco_daughter_deltaR);

fTree->Branch("reco_daughter_dR", &reco_daughter_dR);
fTree->Branch("reco_daughter_to_vertex", &reco_daughter_to_vertex);
fTree->Branch("reco_daughter_slice", &reco_daughter_slice);

fTree->Branch("reco_daughter_shower_to_vertex", &reco_daughter_shower_to_vertex);

fTree->Branch("reco_daughter_shower_startX", &reco_daughter_shower_startX);
fTree->Branch("reco_daughter_shower_startY", &reco_daughter_shower_startY);
fTree->Branch("reco_daughter_shower_startZ", &reco_daughter_shower_startZ);
fTree->Branch("reco_daughter_shower_len", &reco_daughter_shower_len);


fTree->Branch("reco_daughter_PFP_ID", &reco_daughter_PFP_ID);
fTree->Branch("reco_daughter_PFP_nHits", &reco_daughter_PFP_nHits);
fTree->Branch("reco_daughter_PFP_trackScore", &reco_daughter_PFP_trackScore);
fTree->Branch("reco_daughter_PFP_emScore", &reco_daughter_PFP_emScore);
fTree->Branch("reco_daughter_PFP_michelScore", &reco_daughter_PFP_michelScore);
```

reco daughter track

```
fTree->Branch("reco_daughter_trackID", &reco_daughter_trackID);
fTree->Branch("reco_daughter_true_byE_completeness", &reco_daughter_true_byE_completeness);
fTree->Branch("reco_daughter_true_byE_purity", &reco_daughter_true_byE_purity);
fTree->Branch("reco_daughter_true_byE_PDG", &reco_daughter_true_byE_PDG);
fTree->Branch("reco_daughter_true_byE_ID", &reco_daughter_true_byE_ID);
fTree->Branch("reco_daughter_true_byE_origin", &reco_daughter_true_byE_origin);
fTree->Branch("reco_daughter_true_byE_parID", &reco_daughter_true_byE_parID);
fTree->Branch("reco_daughter_true_byE_parPDG", &reco_daughter_true_byE_parPDG);
fTree->Branch("reco_daughter_true_byE_process", &reco_daughter_true_byE_process);

fTree->Branch("reco_daughter_true_byHits_PDG", &reco_daughter_true_byHits_PDG);
fTree->Branch("reco_daughter_true_byHits_ID", &reco_daughter_true_byHits_ID);
fTree->Branch("reco_daughter_true_byHits_origin", &reco_daughter_true_byHits_origin);
fTree->Branch("reco_daughter_true_byHits_parID", &reco_daughter_true_byHits_parID);
fTree->Branch("reco_daughter_true_byHits_parPDG", &reco_daughter_true_byHits_parPDG);
fTree->Branch("reco_daughter_true_byHits_process", &reco_daughter_true_byHits_process);
fTree->Branch("reco_daughter_true_byHits_purity", &reco_daughter_true_byHits_purity);
fTree->Branch("reco_daughter_true_byHits_sharedHits", &reco_daughter_true_byHits_sharedHits);
fTree->Branch("reco_daughter_true_byHits_emHits", &reco_daughter_true_byHits_emHits);

fTree->Branch("reco_daughter_true_byHits_len", &reco_daughter_true_byHits_len);
fTree->Branch("reco_daughter_true_byHits_startX", &reco_daughter_true_byHits_startX);
fTree->Branch("reco_daughter_true_byHits_startY", &reco_daughter_true_byHits_startY);
fTree->Branch("reco_daughter_true_byHits_startZ", &reco_daughter_true_byHits_startZ);
fTree->Branch("reco_daughter_true_byHits_endX", &reco_daughter_true_byHits_endX);
fTree->Branch("reco_daughter_true_byHits_endY", &reco_daughter_true_byHits_endY);
fTree->Branch("reco_daughter_true_byHits_endZ", &reco_daughter_true_byHits_endZ);

fTree->Branch("reco_daughter_true_byHits_startPx", &reco_daughter_true_byHits_startPx);
fTree->Branch("reco_daughter_true_byHits_startPy", &reco_daughter_true_byHits_startPy);
fTree->Branch("reco_daughter_true_byHits_startPz", &reco_daughter_true_byHits_startPz);
fTree->Branch("reco_daughter_true_byHits_startP", &reco_daughter_true_byHits_startP);
fTree->Branch("reco_daughter_true_byHits_startE", &reco_daughter_true_byHits_startE);
//Alternative reco
fTree->Branch("reco_daughter_PFP_true_byHits_PDG", &reco_daughter_PFP_true_byHits_PDG);
fTree->Branch("reco_daughter_PFP_true_byHits_ID", &reco_daughter_PFP_true_byHits_ID);
fTree->Branch("reco_daughter_PFP_true_byHits_origin", &reco_daughter_PFP_true_byHits_origin);
fTree->Branch("reco_daughter_PFP_true_byHits_parID", &reco_daughter_PFP_true_byHits_parID);
fTree->Branch("reco_daughter_PFP_true_byHits_parPDG", &reco_daughter_PFP_true_byHits_parPDG);
fTree->Branch("reco_daughter_PFP_true_byHits_process", &reco_daughter_PFP_true_byHits_process);
fTree->Branch("reco_daughter_PFP_true_byHits_sharedHits",
&reco_daughter_PFP_true_byHits_sharedHits);
fTree->Branch("reco_daughter_PFP_true_byHits_emHits", &reco_daughter_PFP_true_byHits_emHits);

fTree->Branch("reco_daughter_PFP_true_byHits_len", &reco_daughter_PFP_true_byHits_len);
fTree->Branch("reco_daughter_PFP_true_byHits_startX", &reco_daughter_PFP_true_byHits_startX);
fTree->Branch("reco_daughter_PFP_true_byHits_startY", &reco_daughter_PFP_true_byHits_startY);
fTree->Branch("reco_daughter_PFP_true_byHits_startZ", &reco_daughter_PFP_true_byHits_startZ);
fTree->Branch("reco_daughter_PFP_true_byHits_endX", &reco_daughter_PFP_true_byHits_endX);
fTree->Branch("reco_daughter_PFP_true_byHits_endY", &reco_daughter_PFP_true_byHits_endY);
fTree->Branch("reco_daughter_PFP_true_byHits_endZ", &reco_daughter_PFP_true_byHits_endZ);

fTree->Branch("reco_daughter_PFP_true_byHits_startPx", &reco_daughter_PFP_true_byHits_startPx);
fTree->Branch("reco_daughter_PFP_true_byHits_startPy", &reco_daughter_PFP_true_byHits_startPy);
fTree->Branch("reco_daughter_PFP_true_byHits_startPz", &reco_daughter_PFP_true_byHits_startPz);
fTree->Branch("reco_daughter_PFP_true_byHits_startP", &reco_daughter_PFP_true_byHits_startP);
fTree->Branch("reco_daughter_PFP_true_byHits_startE", &reco_daughter_PFP_true_byHits_startE);
fTree->Branch("reco_daughter_PFP_true_byHits_endProcess",
&reco_daughter_PFP_true_byHits_endProcess);
fTree->Branch("reco_daughter_PFP_true_byHits_purity", &reco_daughter_PFP_true_byHits_purity);

fTree->Branch("reco_daughter_allTrack_ID", &reco_daughter_allTrack_ID);
fTree->Branch("reco_daughter_allTrack_dEdX", &reco_daughter_allTrack_dEdX);
fTree->Branch("reco_daughter_allTrack_dQdX", &reco_daughter_allTrack_dQdX);
fTree->Branch("reco_daughter_allTrack_resRange", &reco_daughter_allTrack_resRange);
fTree->Branch("reco_daughter_allTrack_dQdX_SCE", &reco_daughter_allTrack_dQdX_SCE);
fTree->Branch("reco_daughter_allTrack_dEdX_SCE", &reco_daughter_allTrack_dEdX);
fTree->Branch("reco_daughter_allTrack_resRange_SCE", &reco_daughter_allTrack_resRange);

fTree->Branch("reco_daughter_allTrack_calibrated_dEdX", &reco_daughter_allTrack_calibrated_dEdX);
fTree->Branch("reco_daughter_allTrack_calibrated_dEdX_SCE",
&reco_daughter_allTrack_calibrated_dEdX_SCE);

fTree->Branch("reco_daughter_allTrack_Chi2_proton", &reco_daughter_allTrack_Chi2_proton);
fTree->Branch("reco_daughter_allTrack_Chi2_ndof", &reco_daughter_allTrack_Chi2_ndof);

fTree->Branch("reco_daughter_allTrack_Theta", &reco_daughter_allTrack_Theta);
fTree->Branch("reco_daughter_allTrack_Phi", &reco_daughter_allTrack_Phi);

fTree->Branch("reco_daughter_allTrack_len", &reco_daughter_allTrack_len);
```

```
fTree->Branch("reco_daughter_allTrack_startX", &reco_daughter_allTrack_startX);
fTree->Branch("reco_daughter_allTrack_startY", &reco_daughter_allTrack_startY);
fTree->Branch("reco_daughter_allTrack_startZ", &reco_daughter_allTrack_startZ);
fTree->Branch("reco_daughter_allTrack_endX", &reco_daughter_allTrack_endX);
fTree->Branch("reco_daughter_allTrack_endY", &reco_daughter_allTrack_endY);
fTree->Branch("reco_daughter_allTrack_endZ", &reco_daughter_allTrack_endZ);
fTree->Branch("reco_daughter_allTrack_dR", &reco_daughter_allTrack_dR);
fTree->Branch("reco_daughter_allTrack_to_vertex", &reco_daughter_allTrack_to_vertex);
//////

fTree->Branch("reco_daughter_allShower_ID", &reco_daughter_allShower_ID);
fTree->Branch("reco_daughter_allShower_len", &reco_daughter_allShower_len);
fTree->Branch("reco_daughter_allShower_startX", &reco_daughter_allShower_startX);
fTree->Branch("reco_daughter_allShower_startY", &reco_daughter_allShower_startY);
fTree->Branch("reco_daughter_allShower_startZ", &reco_daughter_allShower_startZ);


fTree->Branch("reco_daughter_shower_true_byE_PDG", &reco_daughter_shower_true_byE_PDG);
fTree->Branch("reco_daughter_shower_true_byE_ID", &reco_daughter_shower_true_byE_ID);
fTree->Branch("reco_daughter_shower_true_byE_origin", &reco_daughter_shower_true_byE_origin);
fTree->Branch("reco_daughter_shower_true_byE_parID", &reco_daughter_shower_true_byE_parID);
fTree->Branch("reco_daughter_shower_true_byE_parPDG", &reco_daughter_shower_true_byE_parPDG);

fTree->Branch("reco_daughter_shower_true_byE_startPx", &reco_daughter_shower_true_byE_startPx);
fTree->Branch("reco_daughter_shower_true_byE_startPy", &reco_daughter_shower_true_byE_startPy);
fTree->Branch("reco_daughter_shower_true_byE_startPz", &reco_daughter_shower_true_byE_startPz);
fTree->Branch("reco_daughter_shower_true_byE_startP", &reco_daughter_shower_true_byE_startP);
fTree->Branch("reco_daughter_shower_true_byE_endProcess", &reco_daughter_shower_true_byE_endProcess);


fTree->Branch("reco_daughter_shower_true_byHits_PDG", &reco_daughter_shower_true_byHits_PDG);
fTree->Branch("reco_daughter_shower_true_byHits_ID", &reco_daughter_shower_true_byHits_ID);
fTree->Branch("reco_daughter_shower_true_byHits_origin", &reco_daughter_shower_true_byHits_origin);
fTree->Branch("reco_daughter_shower_true_byHits_parID", &reco_daughter_shower_true_byHits_parID);
fTree->Branch("reco_daughter_shower_true_byHits_parPDG", &reco_daughter_shower_true_byHits_parPDG);
fTree->Branch("reco_daughter_shower_true_byHits_process", &reco_daughter_shower_true_byHits_process);
fTree->Branch("reco_daughter_shower_true_byHits_purity", &reco_daughter_shower_true_byHits_purity);

fTree->Branch("reco_daughter_shower_true_byHits_startPx", &reco_daughter_shower_true_byHits_startPx);
fTree->Branch("reco_daughter_shower_true_byHits_startPy", &reco_daughter_shower_true_byHits_startPy);
fTree->Branch("reco_daughter_shower_true_byHits_startPz", &reco_daughter_shower_true_byHits_startPz);
fTree->Branch("reco_daughter_shower_true_byHits_startP", &reco_daughter_shower_true_byHits_startP);
fTree->Branch("reco_daughter_shower_true_byHits_endProcess", &reco_daughter_shower_true_byHits_endProcess);


///Reconstructed info -- daughter
fTree->Branch("reco_daughter_showerID", &reco_daughter_showerID);
fTree->Branch("reco_daughter_dQdX", &reco_daughter_dQdX);
fTree->Branch("reco_daughter_dEdX", &reco_daughter_dEdX);
fTree->Branch("reco_daughter_resRange", &reco_daughter_resRange);
fTree->Branch("reco_daughter_shower_dQdX", &reco_daughter_shower_dQdX);
fTree->Branch("reco_daughter_shower_dEdX", &reco_daughter_shower_dEdX);
fTree->Branch("reco_daughter_shower_resRange", &reco_daughter_shower_resRange);
fTree->Branch("reco_daughter_len", &reco_daughter_len);
fTree->Branch("reco_daughter_startX", &reco_daughter_startX);
fTree->Branch("reco_daughter_startY", &reco_daughter_startY);
fTree->Branch("reco_daughter_startZ", &reco_daughter_startZ);
fTree->Branch("reco_daughter_endX", &reco_daughter_endX);
fTree->Branch("reco_daughter_endY", &reco_daughter_endY);
fTree->Branch("reco_daughter_endZ", &reco_daughter_endZ);
fTree->Branch("reco_daughter_deltaR", &reco_daughter_deltaR);

fTree->Branch("reco_daughter_dR", &reco_daughter_dR);
fTree->Branch("reco_daughter_to_vertex", &reco_daughter_to_vertex);
fTree->Branch("reco_daughter_slice", &reco_daughter_slice);

fTree->Branch("reco_daughter_shower_to_vertex", &reco_daughter_shower_to_vertex);

fTree->Branch("reco_daughter_shower_startX", &reco_daughter_shower_startX);
fTree->Branch("reco_daughter_shower_startY", &reco_daughter_shower_startY);
fTree->Branch("reco_daughter_shower_startZ", &reco_daughter_shower_startZ);
fTree->Branch("reco_daughter_shower_len", &reco_daughter_shower_len);


fTree->Branch("reco_daughter_PFP_ID", &reco_daughter_PFP_ID);
fTree->Branch("reco_daughter_PFP_nHits", &reco_daughter_PFP_nHits);
fTree->Branch("reco_daughter_PFP_trackScore", &reco_daughter_PFP_trackScore);
fTree->Branch("reco_daughter_PFP_emScore", &reco_daughter_PFP_emScore);
fTree->Branch("reco_daughter_PFP_michelScore", &reco_daughter_PFP_michelScore);
```

reco daughter track

```
fTree->Branch("reco_daughter_trackID", &reco_daughter_trackID);
fTree->Branch("reco_daughter_true_byE_completeness", &reco_daughter_true_byE_completeness);
fTree->Branch("reco_daughter_true_byE_purity", &reco_daughter_true_byE_purity);
fTree->Branch("reco_daughter_true_byE_PDG", &reco_daughter_true_byE_PDG);
fTree->Branch("reco_daughter_true_byE_ID", &reco_daughter_true_byE_ID);
fTree->Branch("reco_daughter_true_byE_origin", &reco_daughter_true_byE_origin);
fTree->Branch("reco_daughter_true_byE_parID", &reco_daughter_true_byE_parID);
fTree->Branch("reco_daughter_true_byE_parPDG", &reco_daughter_true_byE_parPDG);
fTree->Branch("reco_daughter_true_byE_process", &reco_daughter_true_byE_process);

fTree->Branch("reco_daughter_true_byHits_PDG", &reco_daughter_true_byHits_PDG);
fTree->Branch("reco_daughter_true_byHits_ID", &reco_daughter_true_byHits_ID);
fTree->Branch("reco_daughter_true_byHits_origin", &reco_daughter_true_byHits_origin);
fTree->Branch("reco_daughter_true_byHits_parID", &reco_daughter_true_byHits_parID);
fTree->Branch("reco_daughter_true_byHits_parPDG", &reco_daughter_true_byHits_parPDG);
fTree->Branch("reco_daughter_true_byHits_process", &reco_daughter_true_byHits_process);
fTree->Branch("reco_daughter_true_byHits_purity", &reco_daughter_true_byHits_purity);
fTree->Branch("reco_daughter_true_byHits_sharedHits", &reco_daughter_true_byHits_sharedHits);
fTree->Branch("reco_daughter_true_byHits_emHits", &reco_daughter_true_byHits_emHits);

fTree->Branch("reco_daughter_true_byHits_len", &reco_daughter_true_byHits_len);
fTree->Branch("reco_daughter_true_byHits_startX", &reco_daughter_true_byHits_startX);
fTree->Branch("reco_daughter_true_byHits_startY", &reco_daughter_true_byHits_startY);
fTree->Branch("reco_daughter_true_byHits_startZ", &reco_daughter_true_byHits_startZ);
fTree->Branch("reco_daughter_true_byHits_endX", &reco_daughter_true_byHits_endX);
fTree->Branch("reco_daughter_true_byHits_endY", &reco_daughter_true_byHits_endY);
fTree->Branch("reco_daughter_true_byHits_endZ", &reco_daughter_true_byHits_endZ);

fTree->Branch("reco_daughter_true_byHits_startPx", &reco_daughter_true_byHits_startPx);
fTree->Branch("reco_daughter_true_byHits_startPy", &reco_daughter_true_byHits_startPy);
fTree->Branch("reco_daughter_true_byHits_startPz", &reco_daughter_true_byHits_startPz);
fTree->Branch("reco_daughter_true_byHits_startP", &reco_daughter_true_byHits_startP);
fTree->Branch("reco_daughter_true_byHits_startE", &reco_daughter_true_byHits_startE);
//Alternative reco
fTree->Branch("reco_daughter_PFP_true_byHits_PDG", &reco_daughter_PFP_true_byHits_PDG);
fTree->Branch("reco_daughter_PFP_true_byHits_ID", &reco_daughter_PFP_true_byHits_ID);
fTree->Branch("reco_daughter_PFP_true_byHits_origin", &reco_daughter_PFP_true_byHits_origin);
fTree->Branch("reco_daughter_PFP_true_byHits_parID", &reco_daughter_PFP_true_byHits_parID);
fTree->Branch("reco_daughter_PFP_true_byHits_parPDG", &reco_daughter_PFP_true_byHits_parPDG);
fTree->Branch("reco_daughter_PFP_true_byHits_process", &reco_daughter_PFP_true_byHits_process);
fTree->Branch("reco_daughter_PFP_true_byHits_sharedHits",
&reco_daughter_PFP_true_byHits_sharedHits);
fTree->Branch("reco_daughter_PFP_true_byHits_emHits", &reco_daughter_PFP_true_byHits_emHits);

fTree->Branch("reco_daughter_PFP_true_byHits_len", &reco_daughter_PFP_true_byHits_len);
fTree->Branch("reco_daughter_PFP_true_byHits_startX", &reco_daughter_PFP_true_byHits_startX);
fTree->Branch("reco_daughter_PFP_true_byHits_startY", &reco_daughter_PFP_true_byHits_startY);
fTree->Branch("reco_daughter_PFP_true_byHits_startZ", &reco_daughter_PFP_true_byHits_startZ);
fTree->Branch("reco_daughter_PFP_true_byHits_endX", &reco_daughter_PFP_true_byHits_endX);
fTree->Branch("reco_daughter_PFP_true_byHits_endY", &reco_daughter_PFP_true_byHits_endY);
fTree->Branch("reco_daughter_PFP_true_byHits_endZ", &reco_daughter_PFP_true_byHits_endZ);

fTree->Branch("reco_daughter_PFP_true_byHits_startPx", &reco_daughter_PFP_true_byHits_startPx);
fTree->Branch("reco_daughter_PFP_true_byHits_startPy", &reco_daughter_PFP_true_byHits_startPy);
fTree->Branch("reco_daughter_PFP_true_byHits_startPz", &reco_daughter_PFP_true_byHits_startPz);
fTree->Branch("reco_daughter_PFP_true_byHits_startP", &reco_daughter_PFP_true_byHits_startP);
fTree->Branch("reco_daughter_PFP_true_byHits_startE", &reco_daughter_PFP_true_byHits_startE);
fTree->Branch("reco_daughter_PFP_true_byHits_endProcess",
&reco_daughter_PFP_true_byHits_endProcess);
fTree->Branch("reco_daughter_PFP_true_byHits_purity", &reco_daughter_PFP_true_byHits_purity);

fTree->Branch("reco_daughter_allTrack_ID", &reco_daughter_allTrack_ID);
fTree->Branch("reco_daughter_allTrack_dEdX", &reco_daughter_allTrack_dEdX);
fTree->Branch("reco_daughter_allTrack_dQdX", &reco_daughter_allTrack_dQdX);
fTree->Branch("reco_daughter_allTrack_resRange", &reco_daughter_allTrack_resRange);
fTree->Branch("reco_daughter_allTrack_dQdX_SCE", &reco_daughter_allTrack_dQdX_SCE);
fTree->Branch("reco_daughter_allTrack_dEdX_SCE", &reco_daughter_allTrack_dEdX);
fTree->Branch("reco_daughter_allTrack_resRange_SCE", &reco_daughter_allTrack_resRange);

fTree->Branch("reco_daughter_allTrack_calibrated_dEdX", &reco_daughter_allTrack_calibrated_dEdX);
fTree->Branch("reco_daughter_allTrack_calibrated_dEdX_SCE",
&reco_daughter_allTrack_calibrated_dEdX_SCE);

fTree->Branch("reco_daughter_allTrack_Chi2_proton", &reco_daughter_allTrack_Chi2_proton);
fTree->Branch("reco_daughter_allTrack_Chi2_ndof", &reco_daughter_allTrack_Chi2_ndof);

fTree->Branch("reco_daughter_allTrack_Theta", &reco_daughter_allTrack_Theta);
fTree->Branch("reco_daughter_allTrack_Phi", &reco_daughter_allTrack_Phi);

fTree->Branch("reco_daughter_allTrack_len", &reco_daughter_allTrack_len);
```

```
fTree->Branch("reco_daughter_allTrack_startX", &reco_daughter_allTrack_startX);
fTree->Branch("reco_daughter_allTrack_startY", &reco_daughter_allTrack_startY);
fTree->Branch("reco_daughter_allTrack_startZ", &reco_daughter_allTrack_startZ);
fTree->Branch("reco_daughter_allTrack_endX", &reco_daughter_allTrack_endX);
fTree->Branch("reco_daughter_allTrack_endY", &reco_daughter_allTrack_endY);
fTree->Branch("reco_daughter_allTrack_endZ", &reco_daughter_allTrack_endZ);
fTree->Branch("reco_daughter_allTrack_dR", &reco_daughter_allTrack_dR);
fTree->Branch("reco_daughter_allTrack_to_vertex", &reco_daughter_allTrack_to_vertex);
//////

fTree->Branch("reco_daughter_allShower_ID", &reco_daughter_allShower_ID);
fTree->Branch("reco_daughter_allShower_len", &reco_daughter_allShower_len);
fTree->Branch("reco_daughter_allShower_startX", &reco_daughter_allShower_startX);
fTree->Branch("reco_daughter_allShower_startY", &reco_daughter_allShower_startY);
fTree->Branch("reco_daughter_allShower_startZ", &reco_daughter_allShower_startZ);


fTree->Branch("reco_daughter_shower_true_byE_PDG", &reco_daughter_shower_true_byE_PDG);
fTree->Branch("reco_daughter_shower_true_byE_ID", &reco_daughter_shower_true_byE_ID);
fTree->Branch("reco_daughter_shower_true_byE_origin", &reco_daughter_shower_true_byE_origin);
fTree->Branch("reco_daughter_shower_true_byE_parID", &reco_daughter_shower_true_byE_parID);
fTree->Branch("reco_daughter_shower_true_byE_parPDG", &reco_daughter_shower_true_byE_parPDG);

fTree->Branch("reco_daughter_shower_true_byE_startPx", &reco_daughter_shower_true_byE_startPx);
fTree->Branch("reco_daughter_shower_true_byE_startPy", &reco_daughter_shower_true_byE_startPy);
fTree->Branch("reco_daughter_shower_true_byE_startPz", &reco_daughter_shower_true_byE_startPz);
fTree->Branch("reco_daughter_shower_true_byE_startP", &reco_daughter_shower_true_byE_startP);
fTree->Branch("reco_daughter_shower_true_byE_endProcess", &reco_daughter_shower_true_byE_endProcess);


fTree->Branch("reco_daughter_shower_true_byHits_PDG", &reco_daughter_shower_true_byHits_PDG);
fTree->Branch("reco_daughter_shower_true_byHits_ID", &reco_daughter_shower_true_byHits_ID);
fTree->Branch("reco_daughter_shower_true_byHits_origin", &reco_daughter_shower_true_byHits_origin);
fTree->Branch("reco_daughter_shower_true_byHits_parID", &reco_daughter_shower_true_byHits_parID);
fTree->Branch("reco_daughter_shower_true_byHits_parPDG", &reco_daughter_shower_true_byHits_parPDG);
fTree->Branch("reco_daughter_shower_true_byHits_process", &reco_daughter_shower_true_byHits_process);
fTree->Branch("reco_daughter_shower_true_byHits_purity", &reco_daughter_shower_true_byHits_purity);

fTree->Branch("reco_daughter_shower_true_byHits_startPx", &reco_daughter_shower_true_byHits_startPx);
fTree->Branch("reco_daughter_shower_true_byHits_startPy", &reco_daughter_shower_true_byHits_startPy);
fTree->Branch("reco_daughter_shower_true_byHits_startPz", &reco_daughter_shower_true_byHits_startPz);
fTree->Branch("reco_daughter_shower_true_byHits_startP", &reco_daughter_shower_true_byHits_startP);
fTree->Branch("reco_daughter_shower_true_byHits_endProcess", &reco_daughter_shower_true_byHits_endProcess);


///Reconstructed info -- daughter
fTree->Branch("reco_daughter_showerID", &reco_daughter_showerID);
fTree->Branch("reco_daughter_dQdX", &reco_daughter_dQdX);
fTree->Branch("reco_daughter_dEdX", &reco_daughter_dEdX);
fTree->Branch("reco_daughter_resRange", &reco_daughter_resRange);
fTree->Branch("reco_daughter_shower_dQdX", &reco_daughter_shower_dQdX);
fTree->Branch("reco_daughter_shower_dEdX", &reco_daughter_shower_dEdX);
fTree->Branch("reco_daughter_shower_resRange", &reco_daughter_shower_resRange);
fTree->Branch("reco_daughter_len", &reco_daughter_len);
fTree->Branch("reco_daughter_startX", &reco_daughter_startX);
fTree->Branch("reco_daughter_startY", &reco_daughter_startY);
fTree->Branch("reco_daughter_startZ", &reco_daughter_startZ);
fTree->Branch("reco_daughter_endX", &reco_daughter_endX);
fTree->Branch("reco_daughter_endY", &reco_daughter_endY);
fTree->Branch("reco_daughter_endZ", &reco_daughter_endZ);
fTree->Branch("reco_daughter_deltaR", &reco_daughter_deltaR);

fTree->Branch("reco_daughter_dR", &reco_daughter_dR);
fTree->Branch("reco_daughter_to_vertex", &reco_daughter_to_vertex);
fTree->Branch("reco_daughter_slice", &reco_daughter_slice);

fTree->Branch("reco_daughter_shower_to_vertex", &reco_daughter_shower_to_vertex);

fTree->Branch("reco_daughter_shower_startX", &reco_daughter_shower_startX);
fTree->Branch("reco_daughter_shower_startY", &reco_daughter_shower_startY);
fTree->Branch("reco_daughter_shower_startZ", &reco_daughter_shower_startZ);
fTree->Branch("reco_daughter_shower_len", &reco_daughter_shower_len);


fTree->Branch("reco_daughter_PFP_ID", &reco_daughter_PFP_ID);
fTree->Branch("reco_daughter_PFP_nHits", &reco_daughter_PFP_nHits);
fTree->Branch("reco_daughter_PFP_trackScore", &reco_daughter_PFP_trackScore);
fTree->Branch("reco_daughter_PFP_emScore", &reco_daughter_PFP_emScore);
fTree->Branch("reco_daughter_PFP_michelScore", &reco_daughter_PFP_michelScore);
```

reco daughter

```
fTree->Branch("reco_daughter_Chi2_proton", &reco_daughter_Chi2_proton);
fTree->Branch("reco_daughter_Chi2_ndof", &reco_daughter_Chi2_ndof);
fTree->Branch("reco_daughter_momByRange_proton", &reco_daughter_momByRange_proton);
fTree->Branch("reco_daughter_momByRange_muon", &reco_daughter_momByRange_muon);
fTree->Branch("reco_daughter_allTrack_momByRange_proton", &reco_daughter_allTrack_momByRange_proton);
fTree->Branch("reco_daughter_allTrack_momByRange_muon", &reco_daughter_allTrack_momByRange_muon);

fTree->Branch("reco_daughter_shower_Chi2_proton", &reco_daughter_shower_Chi2_proton);
fTree->Branch("reco_daughter_shower_Chi2_ndof", &reco_daughter_shower_Chi2_ndof);

fTree->Branch("reco_daughter_trackScore", &reco_daughter_trackScore);
fTree->Branch("reco_daughter_emScore", &reco_daughter_emScore);
fTree->Branch("reco_daughter_michelScore", &reco_daughter_michelScore);

fTree->Branch("reco_daughter_shower_trackScore", &reco_daughter_shower_trackScore);
fTree->Branch("reco_daughter_shower_emScore", &reco_daughter_shower_emScore);
fTree->Branch("reco_daughter_shower_michelScore", &reco_daughter_shower_michelScore);
```

```
//Looking at reco daughters from the reco beam track
const std::vector<const recob::Track*> trackDaughters = pfpUtil.GetPFParticleDaughterTracks(*particle,evt,fPFParticleTag,fTrackerTag);
const std::vector<const recob::Shower*> showerDaughters = pfpUtil.GetPFParticleDaughterShowers(*particle,evt,fPFParticleTag,fShowerTag);
std::cout << "Beam particle has " << trackDaughters.size() << " track-like daughters and " << showerDaughters.size() << " shower-like daughters." << std::endl;
std::cout << std::endl;
```

```
// Alternative Reconstruction.
//
// Loop over all of the PFParticles associated as daughters.
// Then, check the CNN score (later implement the GNN score)
//
// Also, get the forced-tracking (pandora2) and
// get calorimetry + other info

for( size_t daughterID : particle->Daughters() ){
  const recob::PFParticle * daughterPFP = &(pfpVec->at( daughterID ));
  reco_daughter_PFP_ID.push_back( daughterID );
```

```
try{
  const recob::Shower* pandora2Shower = pfpUtil.GetPFParticleShower( *daughterPFP, evt, fPFParticleTag, "pandora2Shower" );
  std::cout << "pandora2 shower: " << pandora2Shower << std::endl;

  if( pandora2Shower ){
    reco_daughter_allShower_ID.push_back(     pandora2Shower->ID() );
    reco_daughter_allShower_len.push_back(    pandora2Shower->Length() );
    reco_daughter_allShower_startX.push_back( pandora2Shower->ShowerStart().X() );
    reco_daughter_allShower_startY.push_back( pandora2Shower->ShowerStart().Y() );
    reco_daughter_allShower_startZ.push_back( pandora2Shower->ShowerStart().Z() );
  }
  else{
    reco_daughter_allShower_ID.push_back(       -1  );
    reco_daughter_allShower_len.push_back(    -999. );
    reco_daughter_allShower_startX.push_back( -999. );
    reco_daughter_allShower_startY.push_back( -999. );
    reco_daughter_allShower_startZ.push_back( -999. );
  }
```

```
try{
  const recob::Track* pandora2Track = pfpUtil.GetPFParticleTrack( *daughterPFP, evt, fPFParticleTag, "pandora2Track" );
  std::cout << "pandora2 track: " << pandora2Track << std::endl;

  if( pandora2Track ){
    reco_daughter_allTrack_ID.push_back( pandora2Track->ID() );
```

true daughter

```
fTree->Branch("true_daughter_nPi0", &true_daughter_nPi0);
fTree->Branch("true_daughter_nPiPlus", &true_daughter_nPiPlus);
fTree->Branch("true_daughter_nProton", &true_daughter_nProton);
fTree->Branch("true_daughter_nNeutron", &true_daughter_nNeutron);
fTree->Branch("true_daughter_nPiMinus", &true_daughter_nPiMinus);
fTree->Branch("true_daughter_nNucleus", &true_daughter_nNucleus);
```

```
for( int i = 0; i < true_beam_particle->NumberDaughters(); ++i ){
  int daughterID = true_beam_particle->Daughter(i);

  std::cout << "Daughter " << i << " ID: " << daughterID << std::endl;
  auto part = plist[ daughterID ];
  int pid = part->PdgCode();
  true_beam_daughter_PDG.push_back(pid);
  true_beam_daughter_ID.push_back( part->TrackId() );
```

```
if( part->Process().find( "Inelastic" ) != std::string::npos ){
  std::cout << "Inelastic" << std::endl;
  if( pid == 211  ) ++true_daughter_nPiPlus;
  if( pid == -211 ) ++true_daughter_nPiMinus;
  if( pid == 111  ) ++true_daughter_nPi0;
  if( pid == 2212 ) ++true_daughter_nProton;
  if( pid == 2112 ) ++true_daughter_nNeutron;
  if( pid > 2212  ) ++true_daughter_nNucleus;
}
```

My concerns

- Doing analysis in LArSoft violates most previous requirements. But anyway, let's assume we go this way. This are my concerns:
  1. Porting not trivial at all. Many classes and concepts involved. It would require significant changes to the current code in HighLAND. **Validation in protoduneana will take a while**
  2. Systematics propagation should be very, very fast. **It will take a while to optimize the code in protoduneana**
  3. HighLAND produces a output tree with the results of the selection and systematics propagation, and has dedicated drawing tools. At plotting level one can: i) play with the cuts, ii) play with the systematics and change the binning. **Implementing this functionality in protoduneana will take time**
  4. At the end we want to address all previous points in protoduneana, so we will reinvent the wheel