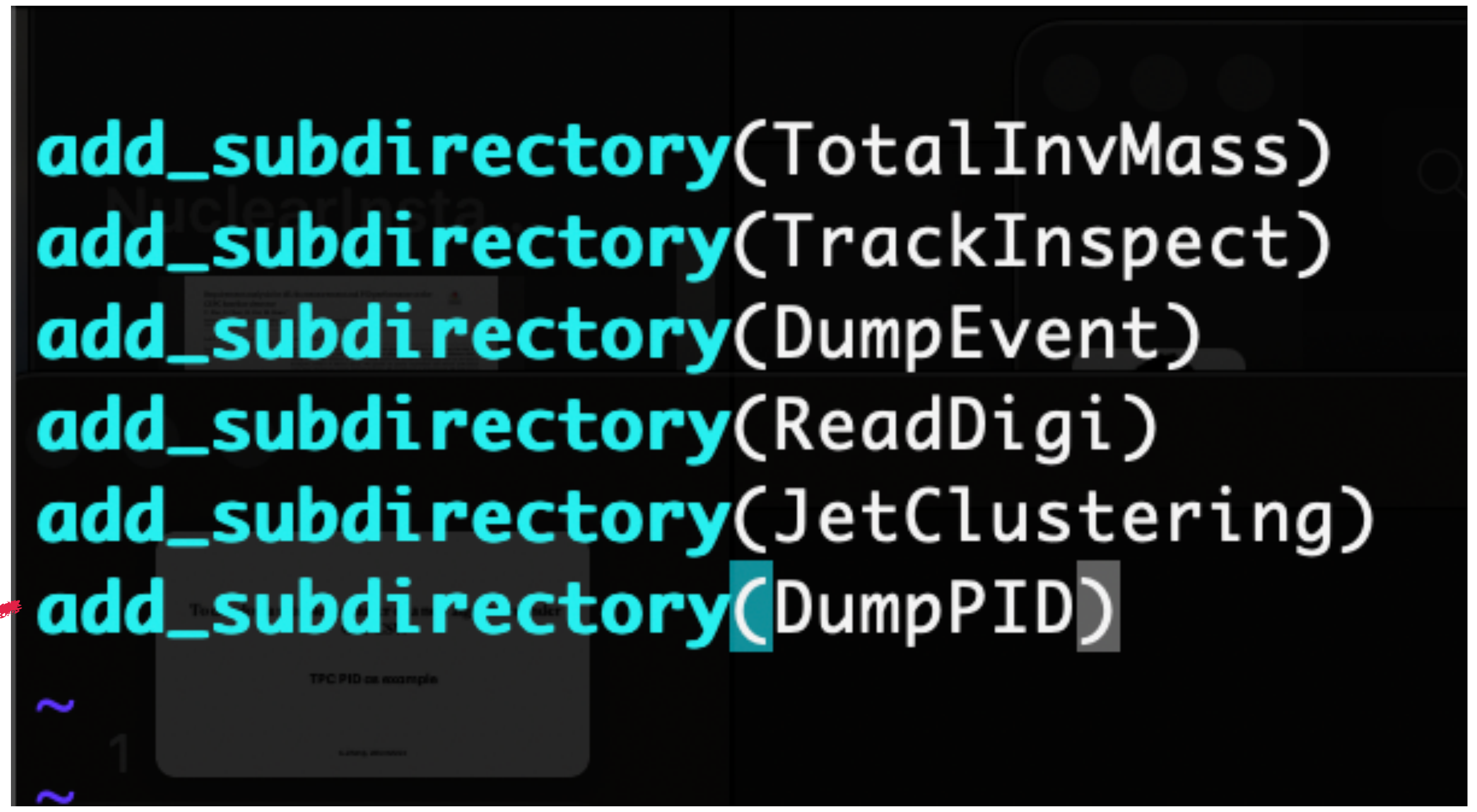
To develop a n-tuple producer or a new algorithm under CEPCSW

TPC PID as an example

C.Zhang, 28Oct2024

Basic structure/syntax for a SW plugin

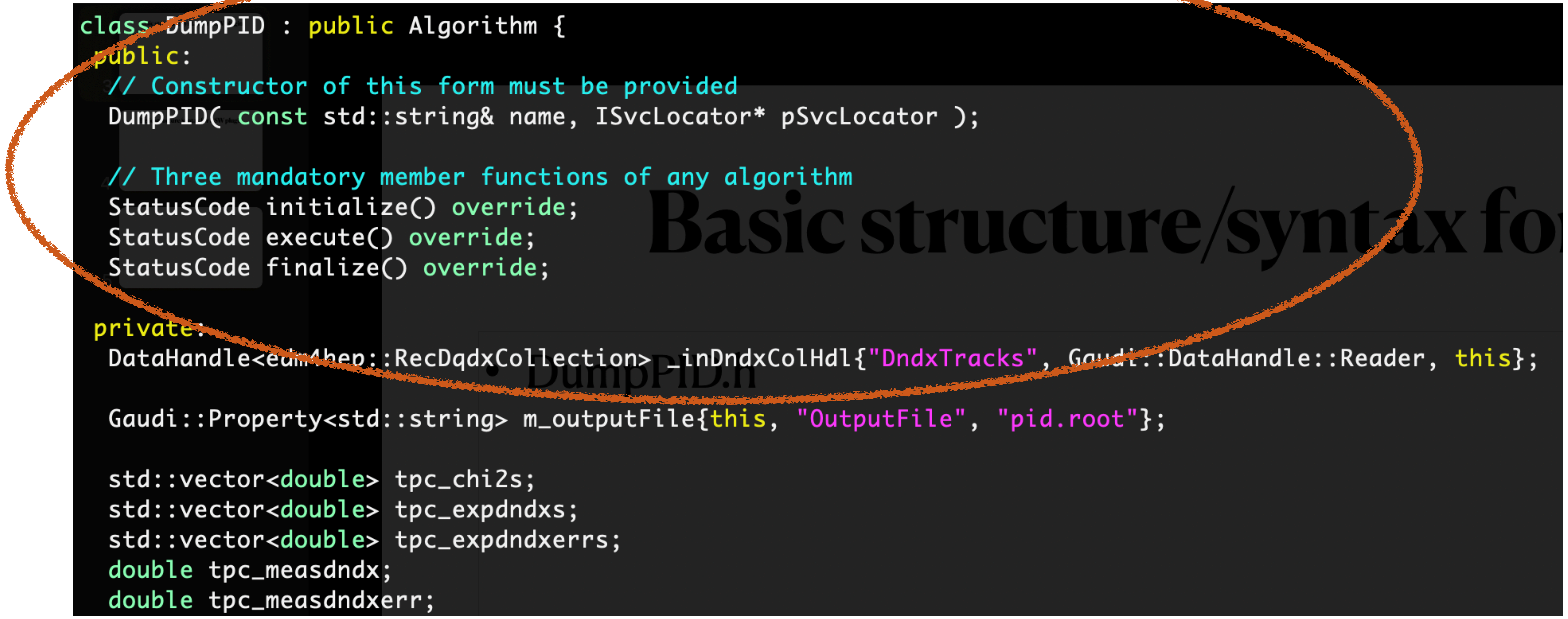
- Under the new SW package we just downloaded
  - cd Analysis/
    - as an exercise we write code here
  - mkdir DumpPID
  - echo add_subdirectory(DumpPID) >> CMakeLists.txt
    - tell SW your package DumPID need to be complied
  - cd DumpPID; touch CMakeLists.txt
    - necessary libs for our code, very common, we can just make a copy from somewhere
    - /publicfs/cms/user/zhangjie/cepc/CEPCSW/Analysis/DumpPID/CMakeLists.txt
  - mkdir src; cd src
  - DumpPID.cpp, DumpPID.h



```
add_subdirectory(TotalInvMass)
add_subdirectory(TrackInspect)
add_subdirectory(DumpEvent)
add_subdirectory(ReadDigi)
add_subdirectory(JetClustering)
add_subdirectory(DumpPID)
~
~
```

Basic structure/syntax for a SW plugin

- DumpPID.h

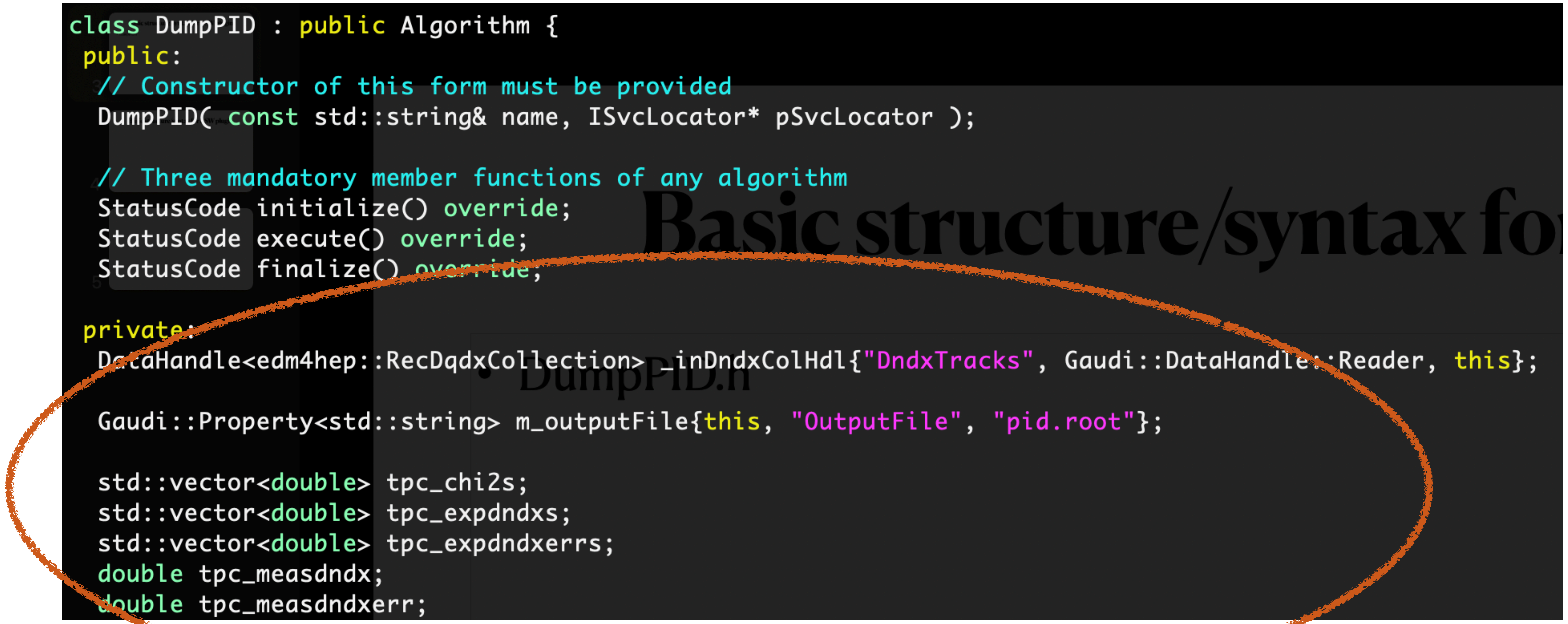


```
class DumpPID : public Algorithm {
 public:
  // Constructor of this form must be provided
  DumpPID( const std::string& name, ISvcLocator* pSvcLocator );

  // Three mandatory member functions of any algorithm
  StatusCode initialize() override;
  StatusCode execute() override;
  StatusCode finalize() override;

 private:
  DataHandle<edm4hep::RecDqdxCollection> _inDndxColHdl{"DndxTracks", Gaudi::DataHandle::Reader, this};

  Gaudi::Property<std::string> m_outputFile{this, "OutputFile", "pid.root"};

  std::vector<double> tpc_chi2s;
  std::vector<double> tpc_expdndxs;
  std::vector<double> tpc_expdndxerrs;
  double tpc_measdndx;
  double tpc_measdndxerr;
```

- Name of the algorithm you are developing
- 3 mandatory member functions just copy/paste

Basic structure/syntax for a SW plugin

- DumpPID.h



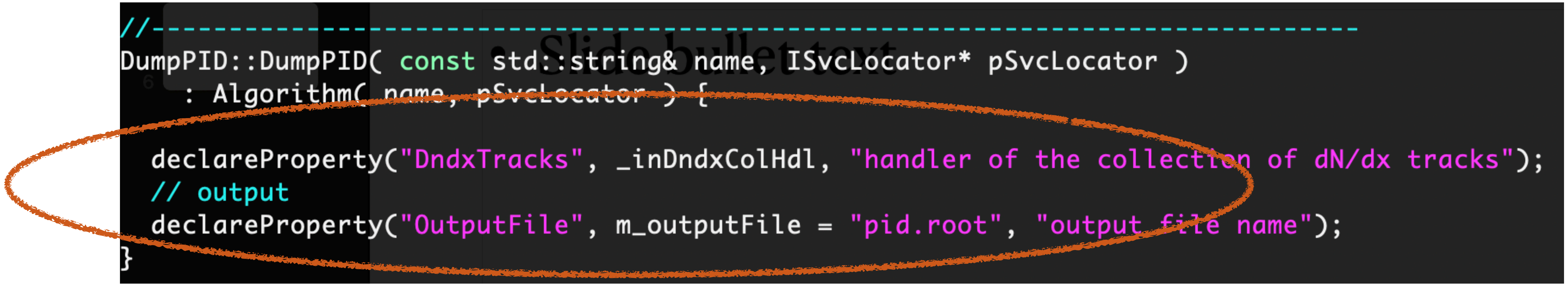
```
class DumpPID : public Algorithm {
 public:
  // Constructor of this form must be provided
  DumpPID( const std::string& name, ISvcLocator* pSvcLocator );

  // Three mandatory member functions of any algorithm
  StatusCode initialize() override;
  StatusCode execute() override;
  StatusCode finalize() override;

 private:
  DataHandle<edm4hep::RecDqdxCollection> _inDndxColHdl{"DndxTracks", Gaudi::DataHandle::Reader, this};

  Gaudi::Property<std::string> m_outputFile{this, "OutputFile", "pid.root"};

  std::vector<double> tpc_chi2s;
  std::vector<double> tpc_expdndxs;
  std::vector<double> tpc_expdndxerrs;
  double tpc_measdndx;
  double tpc_measdndxerr;
```

- The data we will play with in the main function
- float/double/vector
- How to find the dedicated edm4hep::XXXX class?
  - Google
  - $EDM4HEP point out all corresponding sources
  - What info. can be read out from RecDqdxCollection

/cvmfs/cepcsw.ihep.ac.cn/prototype/releases/externals/103.0.2/EDM4hep/include/edm4hep

Basic structure/syntax for a SW plugin

- DumpPID.cpp
- Constructor

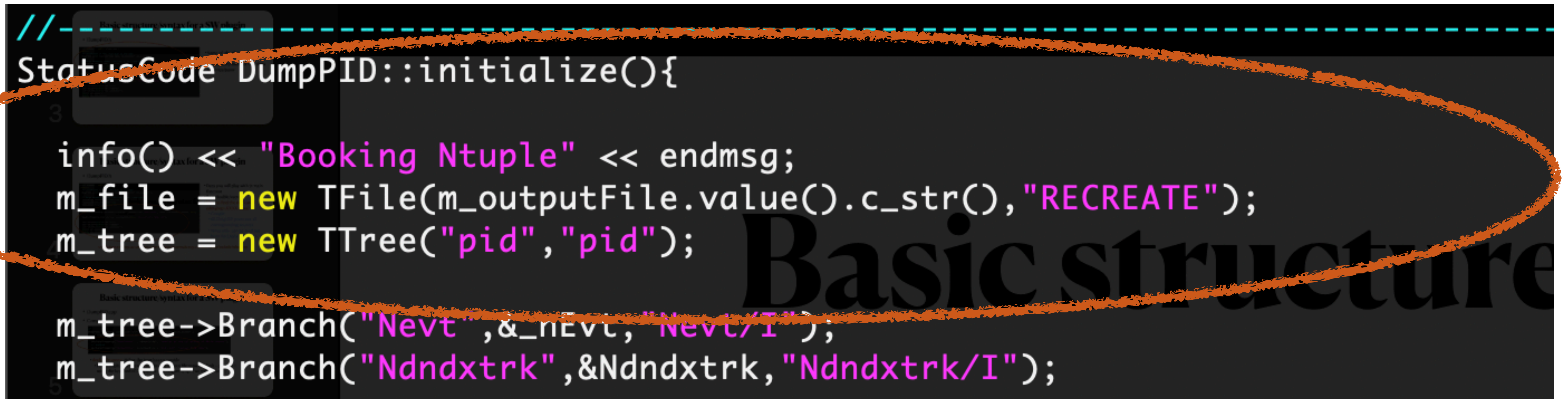


```
//------------------------------------------------------------------------------
DumpPID::DumpPID( const std::string& name, ISvcLocator* pSvcLocator )
  : Algorithm( name, pSvcLocator ) {

  declareProperty("DndxTracks", _inDndxColHdl, "handler of the collection of dN/dx tracks");
  // output
  declareProperty("OutputFile", m_outputFile = "pid.root", "output file name");
}
```

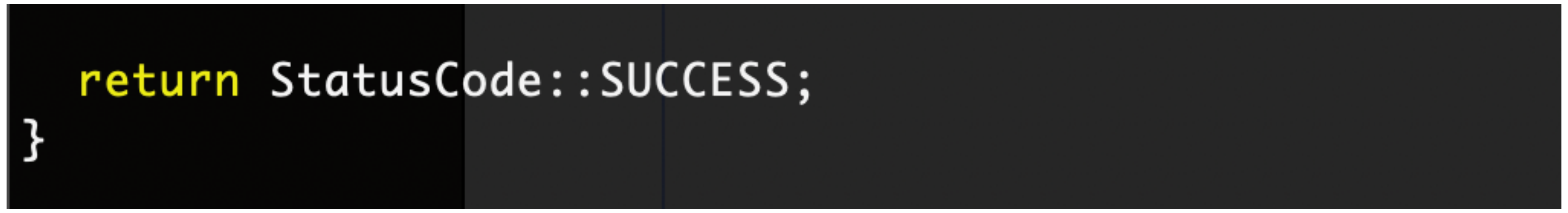
- declareProperty defines the interface of the algo. to outside
  - For example, we can control the output root file name (default ~ pid.root) using the keyword of OutputFile in configure script (we will see this later)

Basic structure/syntax for a SW plugin

- DumpPID.cpp
- Initialize()



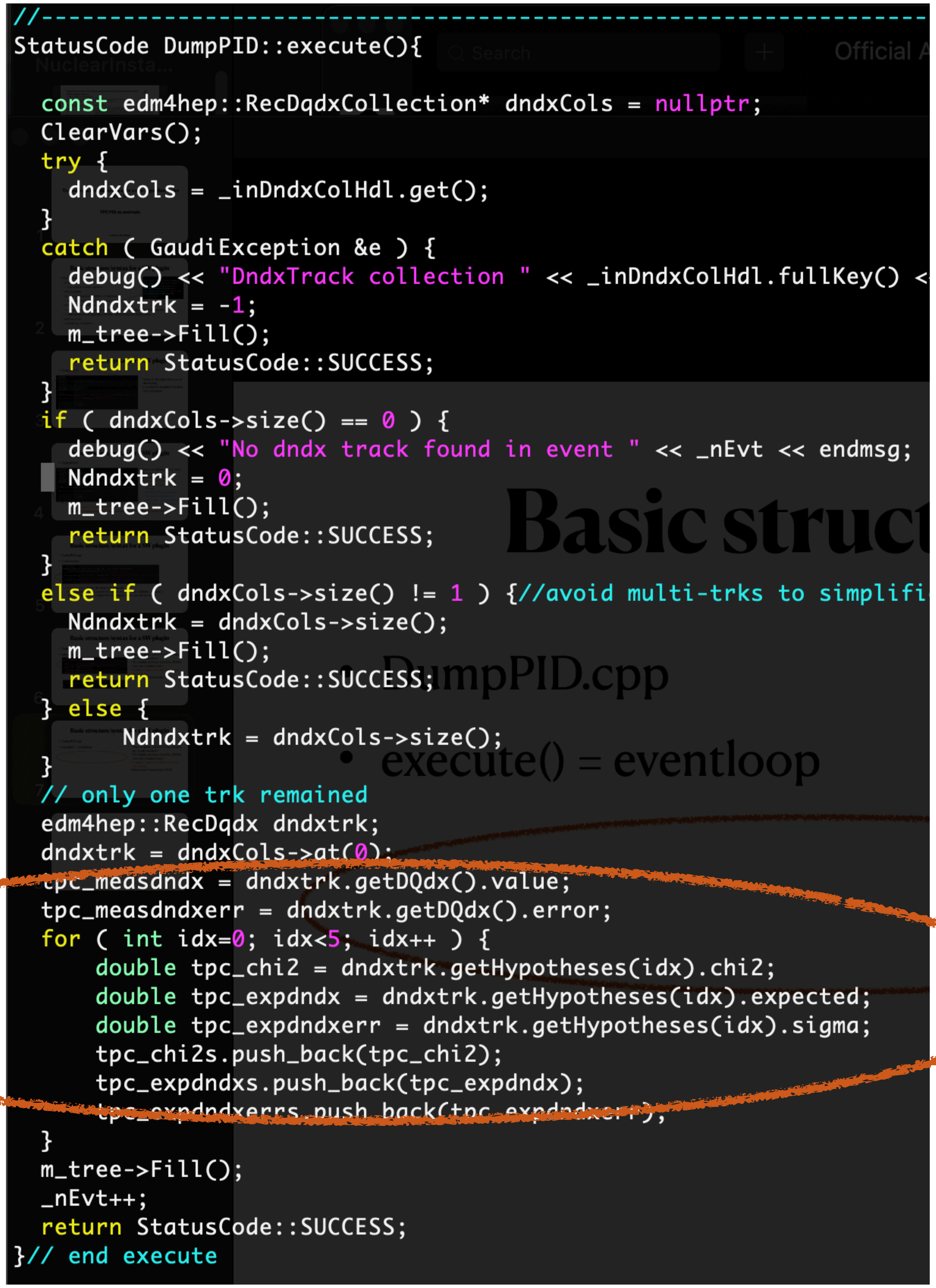
```
//---------------------------------------------------------------------------
StatusCode DumpPID::initialize(){

  info() << "Booking Ntuple" << endmsg;
  m_file = new TFile(m_outputFile.value().c_str(),"RECREATE");
  m_tree = new TTree("pid","pid");

  m_tree->Branch("Nevt",&_nEvt,"Nevt/I");
  m_tree->Branch("Ndndxtrk",&Ndndxtrk,"Ndndxtrk/I");
```



```
  return StatusCode::SUCCESS;
}
```

- Place to define a ROOT::TTree/Branch for Ntuple producer
- To initialise services, GeomSvc, PIDSvc (see the complete code)
- Overall the common futures for all events in the event-loop
- Must return StatusCode::XXXX

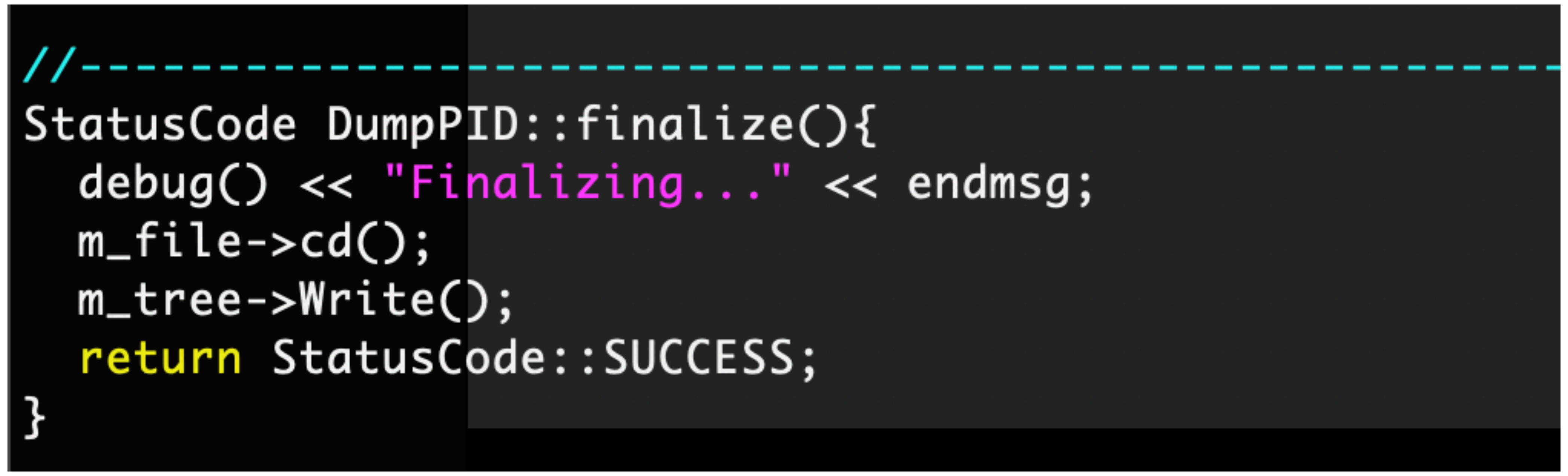
Basic structure/syntax for a SW plugin



```
//----------------------------------------------------------------------
StatusCode DumpPID::execute(){

  const edm4hep::RecDqdxCollection* dndxCols = nullptr;
  ClearVars();
  try {
    dndxCols = _inDndxColHdl.get();
  }
  catch ( GaudiException &e ) {
    debug() << "DndxTrack collection " << _inDndxColHdl.fullKey() <
    Ndndxtrk = -1;
    m_tree->Fill();
    return StatusCode::SUCCESS;
  }
  if ( dndxCols->size() == 0 ) {
    debug() << "No dndx track found in event " << _nEvt << endmsg;
    Ndndxtrk = 0;
    m_tree->Fill();
    return StatusCode::SUCCESS;
  }
  else if ( dndxCols->size() != 1 ) {//avoid multi-trks to simplifi
    Ndndxtrk = dndxCols->size();
    m_tree->Fill();
    return StatusCode::SUCCESS;
  } else {
        Ndndxtrk = dndxCols->size();
  }
  // only one trk remained
  edm4hep::RecDqdx dndxtrk;
  dndxtrk = dndxCols->at(0);
  tpc_measdndx = dndxtrk.getDQdx().value;
  tpc_measdndxerr = dndxtrk.getDQdx().error;
  for ( int idx=0; idx<5; idx++ ) {
      double tpc_chi2 = dndxtrk.getHypotheses(idx).chi2;
      double tpc_expdndx = dndxtrk.getHypotheses(idx).expected;
      double tpc_expdndxerr = dndxtrk.getHypotheses(idx).sigma;
      tpc_chi2s.push_back(tpc_chi2);
      tpc_expdndxs.push_back(tpc_expdndx);
      tpc_expdndxerrs.push_back(tpc_expdndxerr);
  }
  m_tree->Fill();
  _nEvt++;
  return StatusCode::SUCCESS;
}// end execute
```

- DumpPID.cpp
- execute() = eventloop
  - Operate input data event-by-event
  - One can calculate
    - transverse momenta of a track
    - invariant mass of tracks
    - probability of particle types
    - all variables you are interested in.
  - Already know what we can do with edm4hep::RecDqdx by checking its source file
  - Must return StatusCode::XXXX

Basic structure/syntax for a SW plugin

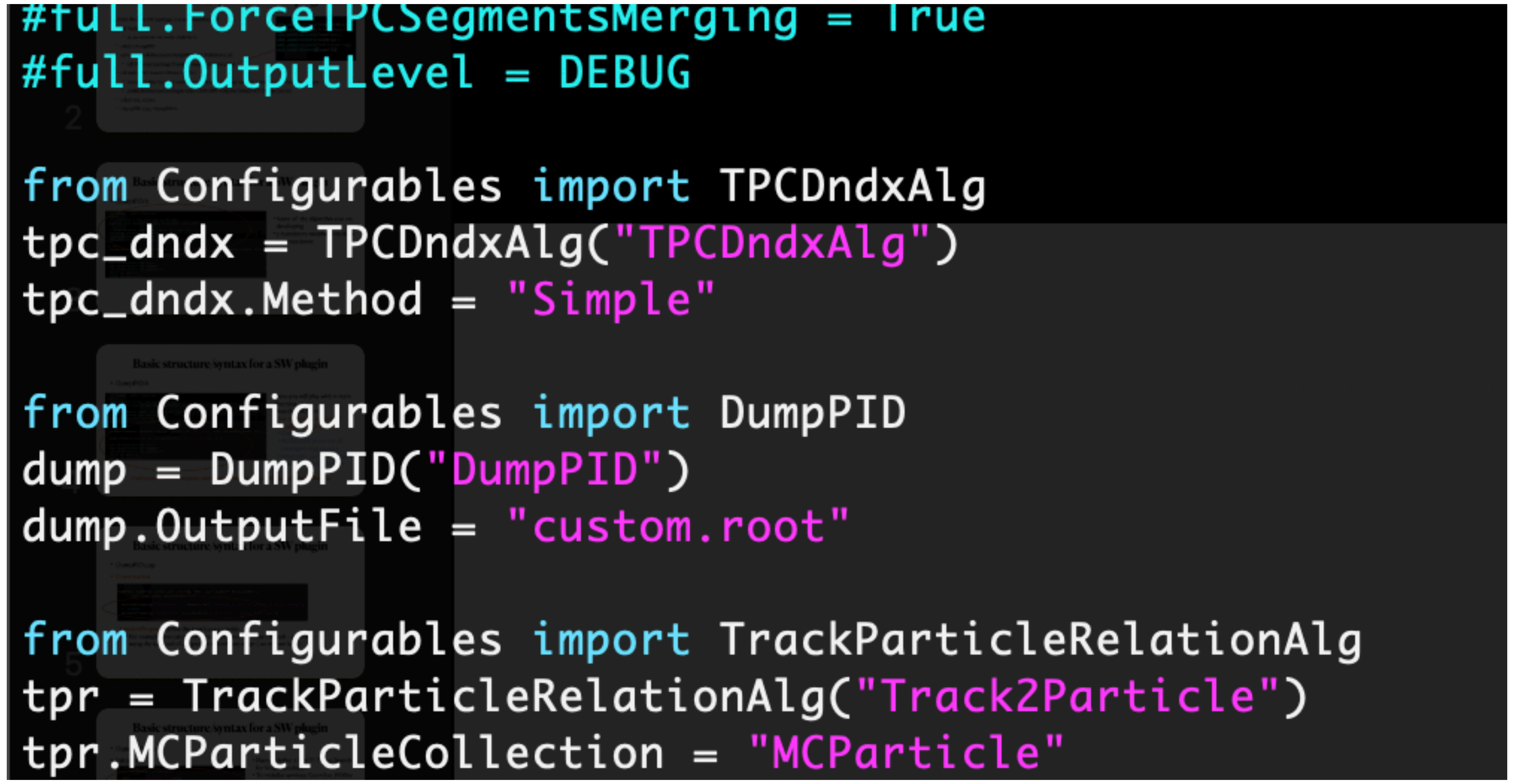


```
//------------------------------------------------------
StatusCode DumpPID::finalize(){
  debug() << "Finalizing..." << endmsg;
  m_file->cd();
  m_tree->Write();
  return StatusCode::SUCCESS;
}
```

- DumpPID.cpp
- finalize()
  - Write out the TTree or do nothing

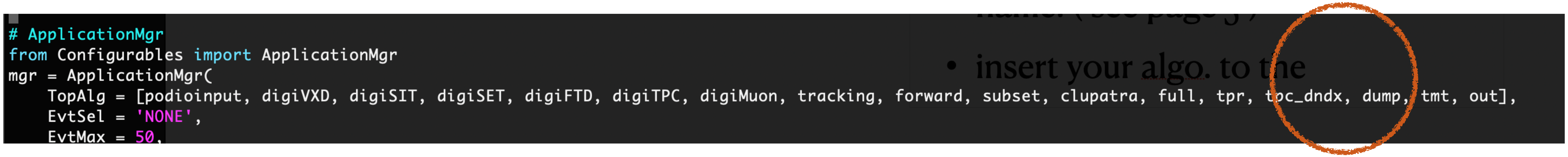
- Copy the DumpPID.cpp and .h to working-area and re-compile your SW
  - /publicfs/cms/user/zhangjie/cepc/CEPCSW/Analysis/DumpPID/src/DumpPID.cpp and .h

Setup your plugin



```
#full.ForceTPCSegmentsMerging = True
#full.OutputLevel = DEBUG

from Configurables import TPCDndxAlg
tpc_dndx = TPCDndxAlg("TPCDndxAlg")
tpc_dndx.Method = "Simple"

from Configurables import DumpPID
dump = DumpPID("DumpPID")
dump.OutputFile = "custom.root"

from Configurables import TrackParticleRelationAlg
tpr = TrackParticleRelationAlg("Track2Particle")
tpr.MCParticleCollection = "MCParticle"
```

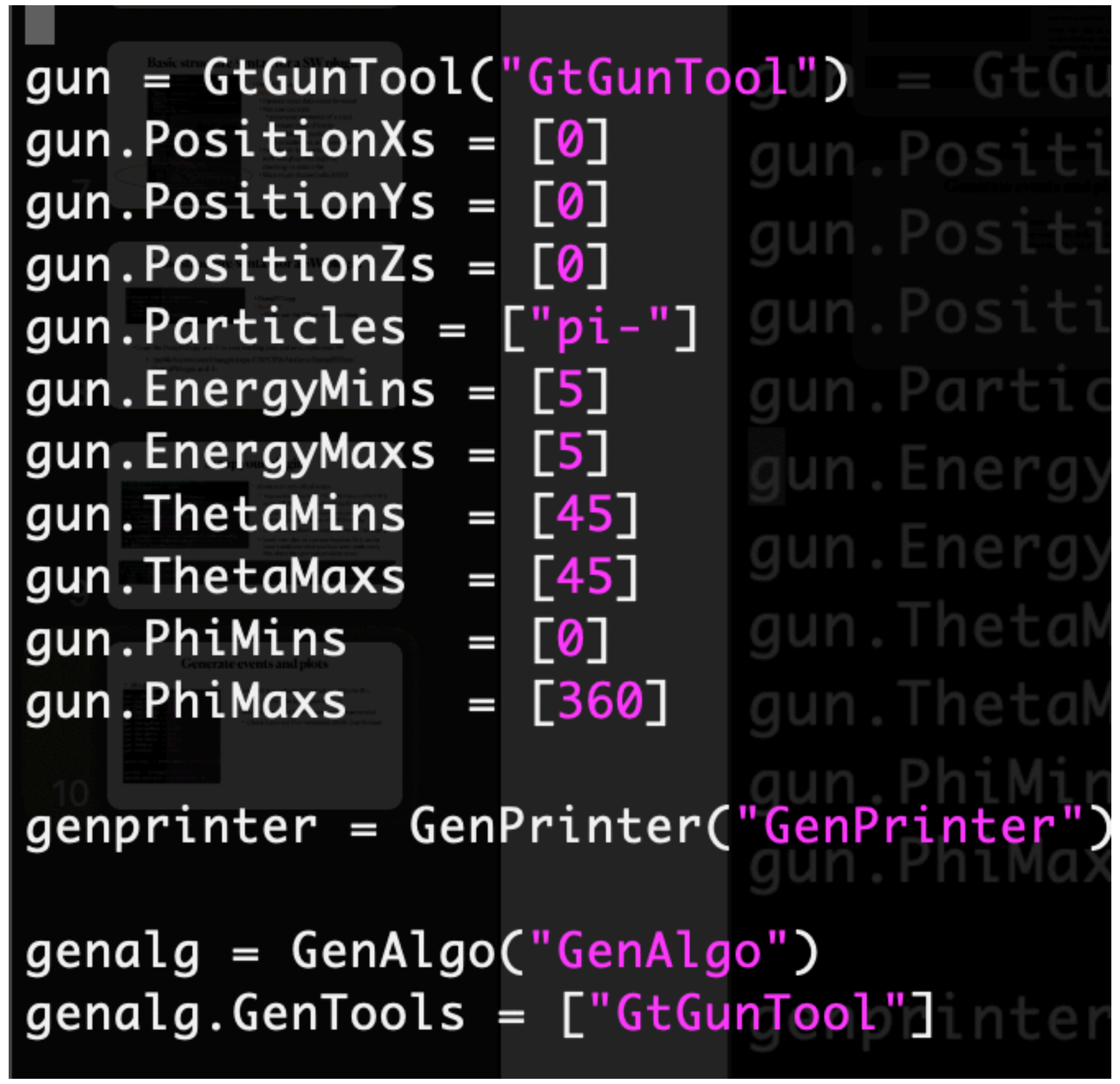
- Should start with official scripts
  - From working directory, CEPCSW/Detector/DetCRD/scripts/TDR_o1_v01/sim.py and tracking.py
  - Need to know what have been produced by upstream. In this exercise, it's TPCDndxAlg
  - We have a interface for output file name. (see page 5)
  - Insert your algo. to a proper location. Or it can be used standalone when you have some made ready files where the upstream products saved.



```
# ApplicationMgr
from Configurables import ApplicationMgr
mgr = ApplicationMgr(
    TopAlg = [podioinput, digiVXD, digiSIT, digiSET, digiFTD, digiTPC, digiMuon, tracking, forward, subset, clupatra, full, tpr, tpc_dndx, dump, tmt, out],
    EvtSel = 'NONE',
    EvtMax = 50,
```

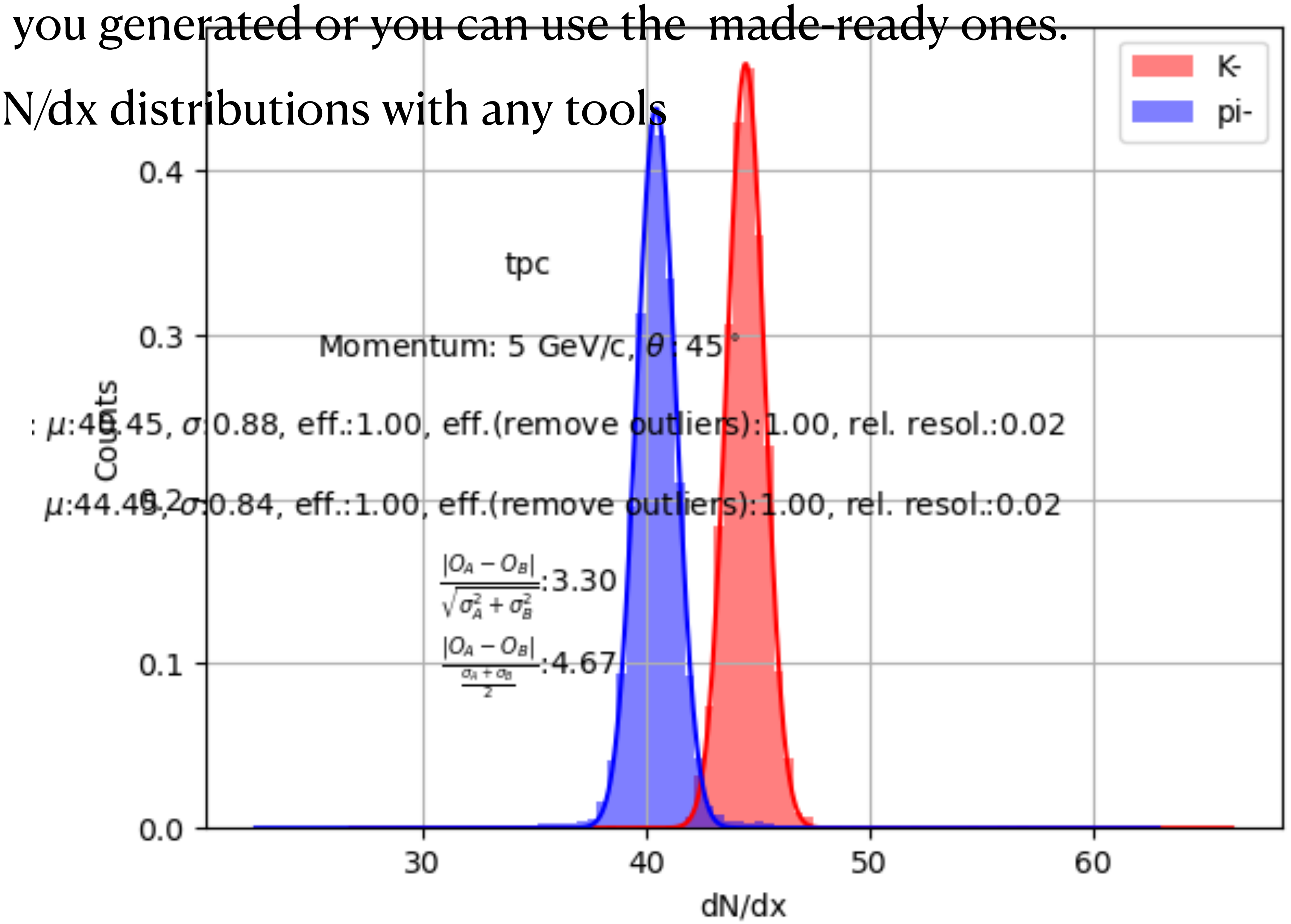
Generate events and plots

- sim.py



```
gun = GtGunTool("GtGunTool")
gun.PositionXs = [0]
gun.PositionYs = [0]
gun.PositionZs = [0]
gun.Particles = ["pi-"]
gun.EnergyMins = [5]
gun.EnergyMaxs = [5]
gun.ThetaMins  = [45]
gun.ThetaMaxs  = [45]
gun.PhiMins    = [0]
gun.PhiMaxs    = [360]

genprinter = GenPrinter("GenPrinter")

genalg = GenAlgo("GenAlgo")
genalg.GenTools = ["GtGunTool"]
```

- Generate some charged Pions (pi-) and Kaons (K-)
  - Find all scripts for job submit and the made-ready root files here
    - /publicfs/cms/user/zhangjie/cepc/CEPCSW/sub_condor.sh and dump_condor.sh
  - Some made-ready files with large statistics
    - /publicfs/cms/user/zhangcg/cepc/fromGZhao/CEPCSW/tuples/anatuples/
- Based on Kaon and Pion custom.root you generated or you can use the made-ready ones.
  - Check Kaon and Pion ionisation dN/dx distributions with any tools



Thanks for your attention

- The complete package of PID in the Reconstruction/ParticleID/ to check something more.
- There are many contributions you can do
  - TPC+ToF PID performance is bad for muon and electron,
  - Primary/secondary vertex reconstruction not available in current SW,
    - A working version is ready, need to check its performance
  - …
  - …
  -