Subject: Re: Kaon lists from PndEventReader::FillList are empty
Posted by Ralf Kliemt on Thu, 02 Feb 2012 12:19:18 GMT
View Forum Message <> Reply to Message

Hi.

I uploaded the incremented version number of PndPidProbability to the svn. I fear that changes in the class introduce the inability to read data produced with an earlier version.

PndAnalysis requires you to set up and initialize a FairRunAna, as you would do it in other macros. E.g. like that:

```
  FairRunAna* fRun = new FairRunAna();
  FairRuntimeDb* rtdb = fRun->GetRuntimeDb();
  TString filetag = "MyTest_sim.root";
  filetag.ReplaceAll("_sim.root",".root");
  PndFileNameCreator namecreator(filetag.Data());
  TString simFile = namecreator.GetSimFileName();
  TString parFile = namecreator.GetParFileName();
  TString recoFile = namecreator.GetRecoFileName();
  TString tracksFile = namecreator.GetCustomFileName("tracks");
  TString pidFile = namecreator.GetCustomFileName("pid");
  TString histoFile = namecreator.GetCustomFileName("histos");
  TString evrdummy = namecreator.GetCustomFileName("evrdummy");
  fRun->SetInputFile(simFile);
  fRun->AddFriend(recoFile);
  fRun->AddFriend(tracksFile);
  fRun->AddFriend(pidFile);
  FairParRootFileIo* parIO = new FairParRootFileIo();
  parIO->open(parFile.Data());
  rtdb->setFirstInput(parIO);
  rtdb->setOutput(parIO);
  fRun->SetOutputFile(evrdummy.Data());
  FairGeane* geane = new FairGeane();
  fRun->AddTask(geane);
  fRun->Init();
```

You can find in macros/mvd/Tools.C a function doing that (with the namecreator): TString InitDefaultRun(TString filetag) While filetag is your *_sim.root file.

Then create a PndAnalysis object, loop over the events and retrieve your lists as always in the eventreader:

```
  PndAnalysis* theAnalysis = new PndAnalysis(); //("SttMvdGemGenTrack","FTSTrkIdeal");
// you can put up to two strings, if you need an a speciic array containing PndTrack objects
  int evts = theAnalysis->GetEntries();
  while ((theAnalysis->GetEvent())&&(ievt++<evts))
  {
    theAnalysis->FillList(mctracks,"McTruth");
    theAnalysis->FillList(pionp,"PionLoosePlus");
```

```
    theAnalysis->FillList(pionm,"PionLooseMinus");
// [... your analysis code ]
  }
```

Now, the thing is that a new interfacing to PID inforation is implemented, and you should be able to do:

```
theAnalysis->FillList(pionp,"PionLoosePlus","PidAlgoMvd;PidAlgoStt;PidAlgoEmcBayes;PidAlgoDrc;PidAlgoDisc;PidAlgoMdtHardCuts");
//or
theAnalysis->FillList(pionp,"PionLoosePlus","PidAlgoIdealCharged"); //(default MC-PID algo)
//or
theAnalysis->FillList(pionp,"PionLoosePlus","PidMvaChargedProbability");//if you tried MVA or other combined PID algorithm
```

It depends what PID Associator task was running (you can run any and all in parallel(!) before. By default there is the PndPidIdealAssociatorTask in the macros, i presume. The names are the TClonesArray names (look into the root files or the classes). The ones separated by semicolon will be combined by Bayes theorem (multiply pdf).

The definition of "VeryLoose", "Loose", "Tight", "VeryTight" is done via the RTDB in macro/params/pidana.par. Be rminded that there is "All" and "Best", too.

Attention: You don't need to use the automatic FillList()! You can create Lists with "Charged" only and select by PID later, using PndAnaPidCombiner to write the desired PID combination to the TCandList (use function Apply()) and use PndAnaPidSelector as selector. This gives a bit more freedom. Both classes are used within FillList of PndAnalysis.

I hope to have helped.

Ralf