New MakeDSelector

Benedikt Zihlmann

February 22, 2021

MakeDSelector

New MakeDSelector in branch "beniNewMkDSelector" of "gluex_root_analysis"

- The usual two files: DSelector_yourname.h and DSelector_yourname.C
- New method: ProcessCombo()
- Move all combo calculations into ProcessCombo()
- Loop over event combos twice
- Allows for reweighing events in second loop.
- Counts and saves statistics for each event

The Header file

New Method: ProcessCombo()

```
class DSelector_etaprimeTEST1 : public DSelector
{
  public:

  DSelector_etaprimeTEST1(TTree* locTree = NULL) :
                          DSelector(locTree){}
  virtual ~DSelector_etaprimeTEST1(){}

  void Init(TTree *tree);
  Bool_t Process(Long64_t entry);
  Bool_t ProcessCombo(Int_t ComboID, Int_t LoopID);

  private:
```

The Header file

Sets of maps of sets

```
// UNIQUENESS TRACKING
set<Int_t> dUsedSoFar_BeamEnergy; //Int_t: Unique ID for beam particles.
set<map<Particle_t, set<Int_t> > > dUsedSoFar_MissingMass;

set<map<Int_t, set<Int_t> > > dEventFS_All_withBEAM; // BEAM is 0
set<map<Int_t, set<Int_t> > > dEventFS_All;
set<map<Int_t, set<Int_t> > > dEventFS_Charged; // 10
set<map<Int_t, set<Int_t> > > dEventFS_Positive; // 1
set<map<Int_t, set<Int_t> > > dEventFS_Negative; // 2
set<map<Int_t, set<Int_t> > > dEventFS_Neutral; // 3
std::pair<std::set<map<Int_t, set<Int_t> > >::iterator,bool> dRetVal;
Int_t dThisEventNumberOfFS;
Int_t dThisEventNumberOfFSAndInTime;
Int_t dThisEventNumberOfFSInTime;
Int_t dThisEventNumberOfFSOutOfTime;
Int_t dThisEventNumberOfChargedFS;
Int_t dThisEventNumberOfPositiveFS;
Int_t dThisEventNumberOfNegativeFS;
Int_t dThisEventNumberOfNeutralFS;

Int_t dEventNChargedTracks;
Int_t dEventNNeutralShowers;
Int_t dEventNBeamPhotons;
```

Method Init()

Initialize root files:

```
    //USERS: SET OUTPUT FILE NAME
    dOutputFileName = "etaprimeTEST1.root";
    dOutputTreeFileName = ""; //"" for none
    dFlatTreeFileName = "FlatTree_etaprimeTEST1.root";
    dFlatTreeName = "";
```

Method Init()

Initialize counters in flat tree:

```
// KEEP STATISTICS FROM EVENT Uniqueness
dFlatTreeInterface->Create_Branch_Fundamental<Int_t>
                ("Unique_EVT_int");
dFlatTreeInterface->Create_Branch_FundamentalArray<Int_t>
                ("Unique_EVT_int_array", "Unique_EVT_int");
```

Method Process()

Called for each event, Initialize event counters:

```
// Initilaize event topology counters
dEventFS_All_withBEAM.clear(); // BEAM is 0
dEventFS_All.clear();
dEventFS_Charged.clear(); // 10
dEventFS_Positive.clear(); // 1
dEventFS_Negative.clear(); // 2
dEventFS_Neutral.clear(); // 3
dThisEventNumberOfFS = 0;
dThisEventNumberOfFSAndInTime = 0;
dThisEventNumberOfFSInTime = 0;
dThisEventNumberOfFSOutOfTime = 0;
dThisEventNumberOfChargedFS = 0;
dThisEventNumberOfPositiveFS = 0;
dThisEventNumberOfNegativeFS = 0;
dThisEventNumberOfNeutralFS = 0;

dEventNChargedTracks = Get_NumChargedHypos();
dEventNNeutralShowers = Get_NumNeutralHypos();
dEventNBeamPhotons = Get_NumBeam();
```

Method Process()

Two loops over combos:

```
//Loop over combos
vector <Int_t> locGoodComboList;
for (UInt_t loc_i = 0; loc_i < Get_NumCombos(); ++loc_i)
{
  Bool_t locThisComboOK = DSelector_etaprimeTEST1::ProcessCombo
                                                    (loc_i, 0);
  if ( locThisComboOK ) {
    locGoodComboList.push_back(loc_i);
  }
}

for (UInt_t loc_i = 0; loc_i < locGoodComboList.size(); ++loc_i)
{
  Bool_t locThisComboOK = DSelector_etaprimeTEST1::ProcessCombo
                                  (locGoodComboList[loc_i], 1);
}
/* log combo statistics for this event */
if (dEventFS_All.size() > 0){
  dFlatTreeInterface->Fill_Fundamental<Int_t>("Unique_EVT_int", 11);
  dFlatTreeInterface->Fill_Fundamental<Int_t>("Unique_EVT_int_array", (Int_t)Get_EventNum
  dFlatTreeInterface->Fill_Fundamental<Int_t>("Unique_EVT_int_array", dEventNChargedTracks
  dFlatTreeInterface->Fill_Fundamental<Int_t>("Unique_EVT_int_array", dEventNNeutralShowe
  dFlatTreeInterface->Fill_Fundamental<Int_t>("Unique_EVT_int_array", dEventNBeamPhotons,
  dFlatTreeInterface->Fill_Fundamental<Int_t>("Unique_EVT_int_array", dThisEventNumberOfFS
  dFlatTreeInterface->Fill_Fundamental<Int_t>("Unique_EVT_int_array", dThisEventNumberOfF
  dFlatTreeInterface->Fill_Fundamental<Int_t>("Unique_EVT_int_array", dThisEventNumberOfFS
  dFlatTreeInterface->Fill_Fundamental<Int_t>("Unique_EVT_int_array", dThisEventNumberOfCh
  dFlatTreeInterface->Fill_Fundamental<Int_t>("Unique_EVT_int_array", dThisEventNumberOfP
  dFlatTreeInterface->Fill_Fundamental<Int_t>("Unique_EVT_int_array", dThisEventNumberOfN
  dFlatTreeInterface->Fill_Fundamental<Int_t>("Unique_EVT_int_array", dThisEventNumberOfN
  Fill_FlatTree();
}
```

Method ProcessCombo()

Setup combo, prepare local maps and sets:

```
Int_t loc_i = ComboID;
dComboWrapper->Set_ComboIndex(loc_i);

double locGlobalWeightFactor = 1.;
if (LoopID > 0) { // second call adjust GlobalWeight factor
  if (dThisEventNumberOfFSAndInTime>0)
  locGlobalWeightFactor = 1./(float)dThisEventNumberOfFSAndInTime;
}

map<Int_t, set<Int_t> > locComboFS_All_withBEAM;
map<Int_t, set<Int_t> > locComboFS_All;
map<Int_t, set<Int_t> > locComboFS_Charged;     // 10
map<Int_t, set<Int_t> > locComboFS_Positive;    // 1
map<Int_t, set<Int_t> > locComboFS_Negative;    // 2
map<Int_t, set<Int_t> > locComboFS_Neutral;     // 3
map<Int_t, set<Int_t> > locComboFS_BEAM;        // 0
set<Int_t> locCombo_Neutral;     // all FS neutrals
set<Int_t> locCombo_Charged;     // all FS charged tracks
set<Int_t> locCombo_Positive;    // all FS positive charged tracks
set<Int_t> locCombo_Negative;    // all FS negative charged tracks
set<Int_t> locCombo_BeamPhoton; // current beam photon of this combo

Double_t locKinFitChi2 = dComboWrapper->Get_ChiSq_KinFit();
Double_t locKinFitNDF = dComboWrapper->Get_NDF_KinFit();
Double_t locChi2NDF = locKinFitChi2/locKinFitNDF;
```

Method ProcessCombo()

Method ProcessCombo()

Method ProcessCombo()

Adjust weight, fill first loop histograms

```
// use the following weight for the current combo!
double locTotalComboWeight = locHistAccidWeightFactor * locGlobalWeightFactor ;
double Weight = locTotalComboWeight;

.....
 if ((ALLYOURCUTSOK) && (!LoopID)) {
   // first loop regular Weights
   .....
}

....
if (LoopID){
  //second loop adjusted Wieghts;
  .....
}
```

Method ProcessCombo()

cont.:

```
....
//Bool_t locAllCutsOK = false;
Bool_t locAllCutsOK = ALLYOURCUTSOK;

if ( (locAllCutsOK) && (LoopID == 0) ) {
  locComboFS_Charged.insert(std::make_pair(10, locCombo_Charged));
  locComboFS_Positive.insert(std::make_pair(1, locCombo_Positive));
  locComboFS_Negative.insert(std::make_pair(2, locCombo_Negative));
  locComboFS_Neutral.insert(std::make_pair(3, locCombo_Neutral));
  locComboFS_BEAM.insert(std::make_pair(0, locCombo_BeamPhoton));

  locComboFS_All.insert(std::make_pair(1, locCombo_Positive));
  locComboFS_All.insert(std::make_pair(2, locCombo_Negative));
  locComboFS_All.insert(std::make_pair(3, locCombo_Neutral));

  locComboFS_All_withBEAM.insert(std::make_pair(1, locCombo_Positive));
  locComboFS_All_withBEAM.insert(std::make_pair(2, locCombo_Negative));
  locComboFS_All_withBEAM.insert(std::make_pair(3, locCombo_Neutral));
  locComboFS_All_withBEAM.insert(std::make_pair(0, locCombo_BeamPhoton));


  .....
}

.....
return locAllCutsOK;
```

Method ProcessCombo()