GlueX Analysis Workshop

Session 2c
Analysis Path With DSelectors
Lawrence Ng
Summer 2022

Reaction of Interest $\gamma p \rightarrow (\eta \rightarrow \gamma_3\gamma_4)(\pi^0 \rightarrow \gamma_1\gamma_2)p$

Beam Photons x Spectroscopic Combos = Total Combos

Kinematic Fitter

Adds Kin Fit 4-vectors

Fit must converge

Trees were produced from the ReactionFilter

- pi0eta__B4_M7_M17_Tree
  - B4 = 4 beam bunches on each side of the prompt peak
  - M7/17 = mass of π^0/η unconstrained but photons paired such that $\{\gamma_1, \gamma_2\} \Rightarrow \pi^0$ and $\{\gamma_3, \gamma_4\} \Rightarrow \eta$ in a "loose" mass window

Selecting good events

Lots of combinations of initial+final state particles possible - how to choose the correct one?

Selecting good final state particles

1. Selecting good NeutralParticleHypothesis (i.e. photons)
2. Selecting good ChargedTrackHypothesis (i.e. proton)
3. Selecting exclusive reaction
4. Selecting the kinematics of interest

Handling combinatorics

1. Event counting (uniqueness tracking / weighting)

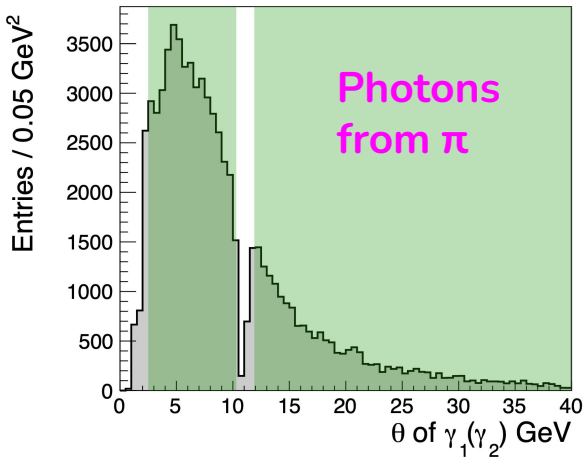
Prepping for further amplitude analysis

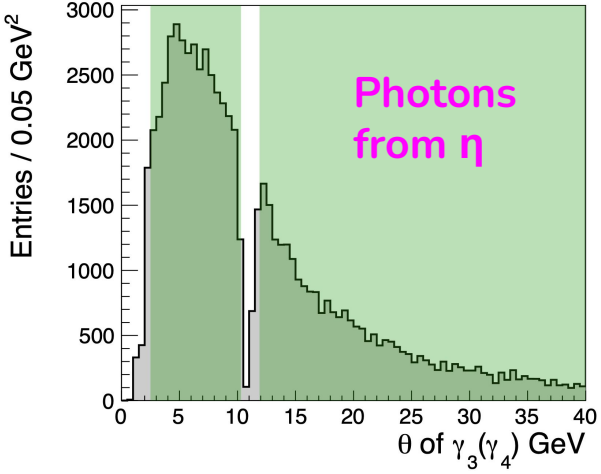
Inside the DSelector

DSelector_etapi.C/h

Lets select good combinations

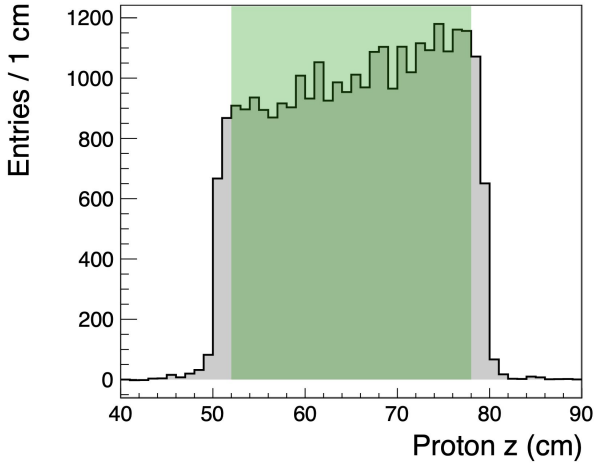
Selecting Good Photons

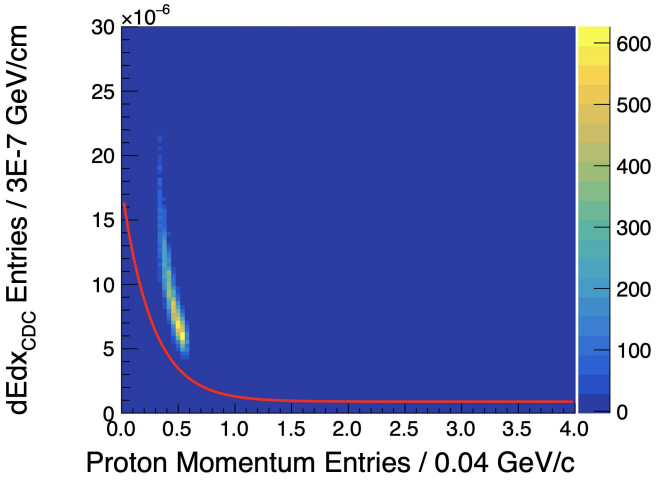





```
// Low energy photons are more likely to be spurious, require a minimum E
bool bPhotonE=(locPhoton1P4.E()>0.1)*(locPhoton2P4.E()>0.1)*(locPhoton3P4.E()>0.1)*(locPhoton4P4.E()>0.1);
// Working in degrees instead of radians, we remove photons near the beamline (<~2.5) and near the BCAL/FCAL transition (<~11.9, >~10.3)
float radToDeg=180/3.14159;
bool bPhotonTheta=
        ((locPhoton1P4.Theta()*radToDeg>=2.5 && locPhoton1P4.Theta()*radToDeg<=10.3) || locPhoton1P4.Theta()*radToDeg>=11.9)*
        ((locPhoton2P4.Theta()*radToDeg>=2.5 && locPhoton2P4.Theta()*radToDeg<=10.3) || locPhoton2P4.Theta()*radToDeg>=11.9)*
        ((locPhoton3P4.Theta()*radToDeg>=2.5 && locPhoton3P4.Theta()*radToDeg<=10.3) || locPhoton3P4.Theta()*radToDeg>=11.9)*
        ((locPhoton4P4.Theta()*radToDeg>=2.5 && locPhoton4P4.Theta()*radToDeg<=10.3) || locPhoton4P4.Theta()*radToDeg>=11.9);
```

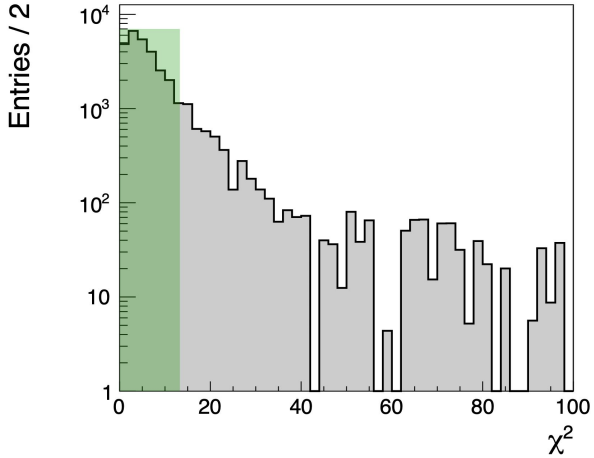
Selecting Good Protons

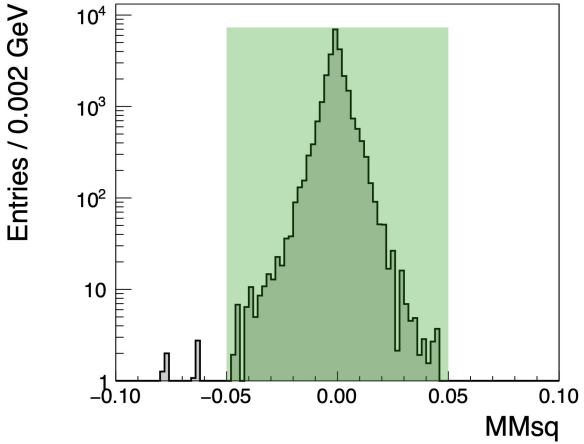





```
// protons need some momentum be reconstructed properly
bool bProtonMomentum=locProtonP4.Vect().Mag()>0.3;
// separate proton/pi+ based on energy loss in CDC
bool bProton_dEdx=dProtonWrapper->Get_dEdx_CDC()>=TMath::Power(10,-6)*(0.9+TMath::Exp(3.0-3.5*(locProtonP4.Vect().Mag()+0.05)/.93827));
// require proton to come from ~ the target region [52,78]centimeters
bool bProtonZ = 52 <= locProtonX4.Z() && locProtonX4.Z() <= 78;
```

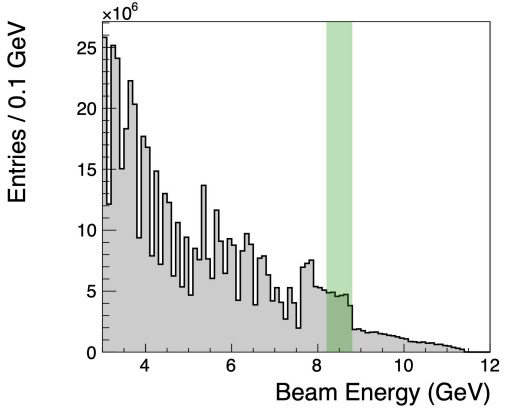
Exclusivity Selections

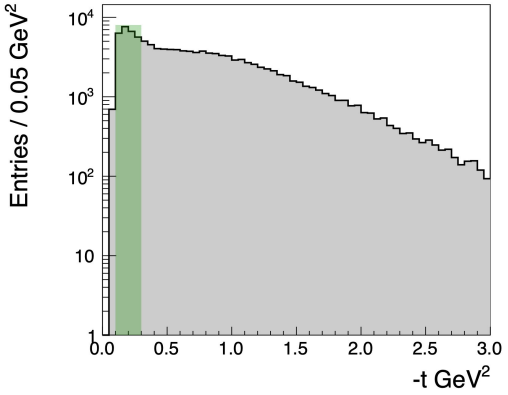


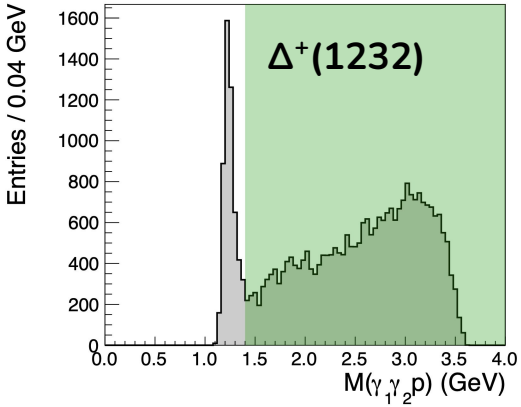



```
// Kinematic fit for this tree only attempts to quantify how well conservation of 4-momentum is maintained
//      4 NDF in this fit where 13.277 corresponds to a p=0.01
bool bChiSq=dComboWrapper->Get_ChiSq_KinFit("")<13.277;
// Unused energy is the sum of unused neutral shower energy not used by this current combo
//      require no unused energy meaning the event only has 4 neutral shower hypotheses to limit final state combinatorics
bool bUnusedEnergy=dComboWrapper->Get_Energy_UnusedShowers()<0.05;
// No missing particles are expected = no missing mass
bool bMMsq=abs(locMissingMassSquared)<0.05;
```

Selecting Relevant Kinematics








```
// Select on coherent peak for region of high polarization. The AMPTOOLS fit using Zlm amplitudes will use the polarization
//      for extra separation power (will tell us something about the production mechanism)
bool bBeamEnergy=(locBeamP4.E()>8.2)*(locBeamP4.E()<8.8);
float mandelstam_t=-(dTargetP4-locProtonP4).M2();
float Mpi0p=(locPhoton1P4+locPhoton2P4+locProtonP4).M();
float Metap=(locPhoton3P4+locPhoton4P4+locProtonP4).M();
float Metapi0=(locPhoton1P4+locPhoton2P4+locPhoton3P4+locPhoton4P4).M();
// Meson production occurs with small-t whereas baryon production occurs with large-t. This analysis cares about mesons
bool bmandelstamt=(mandelstam_t<0.3)*(mandelstam_t>0.1);
// There are a few baryon resonances, the Delta+(1232) being the largest. We can reject it
bool bMpi0p=Mpi0p>1.4;
```

Now we have a good set of potential combinations

How to select the correct combination?

- Event counting
  - Uniqueness tracking (won’t talk about it here)
  - Weighting

Accidental Subtraction

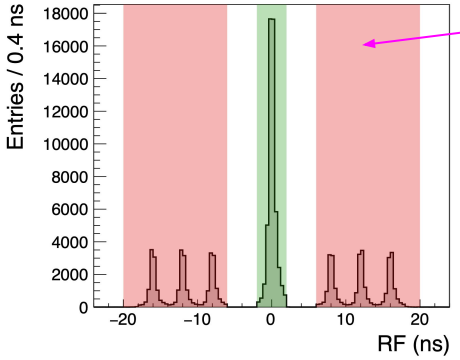
High intensity photon beam comes in bunches ⇒ can have multiple prompt beam photons / event

- Which prompt beam photon initiated the reaction?

Out-of-time beam photons should look similar to non-initial prompt beam photons

- Fold in these “accidentally tagged” beam photons and statistically subtract

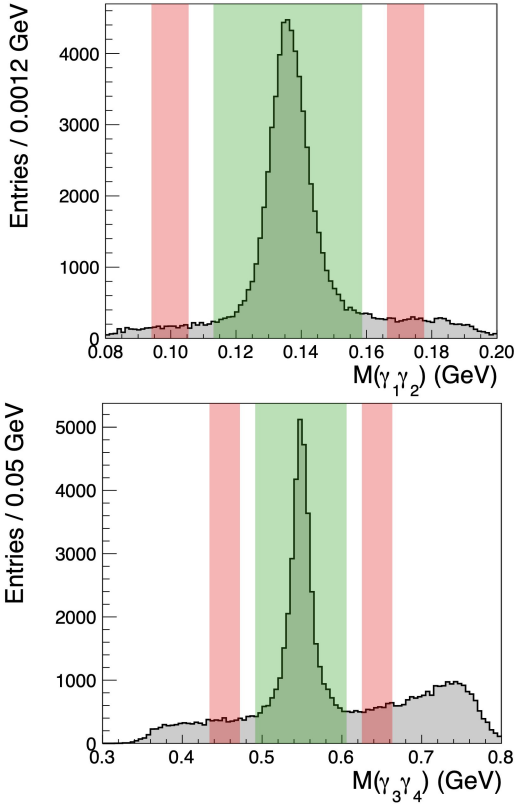
Need energy+run dependent weight scaling ~ lookup table




```
TLorentzVector locBeamX4_Measured = dComboBeamWrapper->Get_X4_Measured();
Double_t locBunchPeriod = dAnalysisUtilities.Get_BeamBunchPeriod(Get_RunNumber());
Double_t locDeltaT_RF = dAnalysisUtilities.Get_DeltaT_RF(Get_RunNumber(), locBeamX4_Measured, dComboWrapper);
// 0 for in-time events, non-zero integer for out-of-time photons
Int_t locRelBeamBucket = dAnalysisUtilities.Get_RelativeBeamBucket(Get_RunNumber(), locBeamX4_Measured, dComboWrapper);
Int_t locNumOutOfTimeBunchesInTree = 4; //YOU need to specify this number
//Number of out-of-time beam bunches in tree (on a single side, so that total number out-of-time bunches accepted is 2 times
//      this number for left + right bunches)
Bool_t locSkipNearestOutOfTimeBunch = true; // True: skip events from nearest out-of-time bunch on either side (recommended).
Int_t locNumOutOfTimeBunchesToUse = locSkipNearestOutOfTimeBunch ? locNumOutOfTimeBunchesInTree-1:locNumOutOfTimeBunchesInTree;
// Ideal value would be 1, but deviations require added factor, which is different for data and MC.
Double_t locAccidentalScalingFactor = dAnalysisUtilities.Get_AccidentalScalingFactor(Get_RunNumber(), locBeamP4.E(), dIsMC);
Double_t locAccidentalScalingFactorError = dAnalysisUtilities.Get_AccidentalScalingFactorError(Get_RunNumber(), locBeamP4.E());
// Weight by 1 for in-time events, ScalingFactor*(1/NBunches) for out-of-time
Double_t locHistAccidWeightFactor = locRelBeamBucket==0 ? 1 : -locAccidentalScalingFactor/(2*locNumOutOfTimeBunchesToUse) ;
if(locSkipNearestOutOfTimeBunch && abs(locRelBeamBucket)==1) { // Skip nearest out-of-time bunch: tails of in-time distribution also leak in
        dComboWrapper->Set_IsComboCut(true);
        continue;
}
```

B4 ReactionFilter from slide 2

Spectroscopic photon combinatorics



For example

Combo 1: $(\eta \to \gamma_3\gamma_4)(\pi^0 \to \gamma_1\gamma_2)$

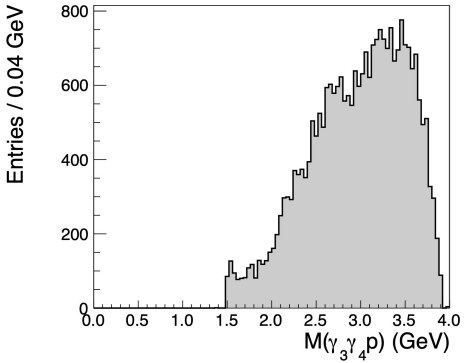
Combo 2: $(\eta \to \gamma_1\gamma_4)(\pi^0 \to \gamma_3\gamma_2)$

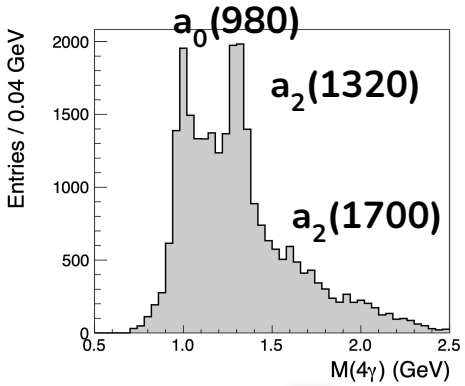
Correct combo should be peaking

```
float Mpi0=locPi0P4.M();
float Meta=locEtaP4.M();
float pi0Mean=0.135881;
float etaMean=0.548625;
float pi0Std=0.0076;
float etaStd=0.0191;
float pi0_sbweight;
float eta_sbweight;
// The signal regions are both +/- 3 sigmas around the peak the left and right sidebands
//      which are some N sigmas wide with some M sigma skip region included
//      between the signal and sideband regions. The weight = the ratio the lengths
//      spanned by the signal to that of the sideband times -1.
if (Mpi0 > pi0Mean-3*pi0Std && Mpi0 < pi0Mean+3*pi0Std){ pi0_sbweight=1; }
else if (Mpi0 > pi0Mean+4*pi0Std && Mpi0 < pi0Mean+5.5*pi0Std){ pi0_sbweight=-2.0; }
else if (Mpi0 > pi0Mean-5.5*pi0Std && Mpi0 < pi0Mean-4*pi0Std){ pi0_sbweight=-2.0; }
else { pi0_sbweight=0; }
if (Meta > etaMean-3*etaStd && Meta < etaMean+3*etaStd){ eta_sbweight=1; }
else if (Meta > etaMean+4*etaStd && Meta < etaMean+6*etaStd){ eta_sbweight=-1.5; }
else if (Meta > etaMean-6*etaStd && Meta < etaMean-4*etaStd){ eta_sbweight=-1.5; }
else { eta_sbweight=0; }
```

Final weight = multiply all your weights

More Variables of Interest





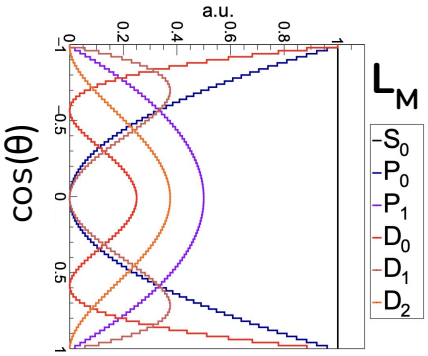
More mass plots

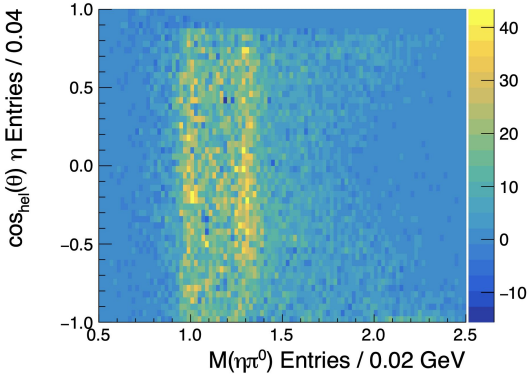
```
float Metap=(locPhoton3P4+locPhoton4P4+locProtonP4).M();
float Metapi0=(locPhoton1P4+locPhoton2P4+locPhoton3P4+locPhoton4P4).M();
```

Decay angular distribution

```
TLorentzVector pi0 = locPhoton1P4+locPhoton2P4;
TLorentzVector eta = locPhoton3P4+locPhoton4P4;
// First we need to boost to center of mass frame
TLorentzRotation cmRestBoost( -(locBeamP4+dTargetP4).BoostVector() );
TLorentzVector pi0_cm = cmRestBoost * pi0;
TLorentzVector eta_cm = cmRestBoost * eta;
TLorentzVector beam_cm = cmRestBoost * locBeamP4;
TLorentzVector recoil_cm = cmRestBoost * locProtonP4;
TLorentzVector resonance = pi0_cm + eta_cm;
// We boost again, now to the resonances rest frame
TLorentzRotation resRestBoost( -resonance.BoostVector() );
TLorentzVector beam_res   = resRestBoost * beam_cm;
TLorentzVector recoil_res = resRestBoost * recoil_cm;
TLorentzVector eta_res = resRestBoost * eta_cm;
//// Redefinition of the axes
TVector3 z = -1. * recoil_res.Vect().Unit(); // Helicity frame
// TVector3 z = beam_res.Vect().Unit(); // Gottfried-Jackson frame
// normal to the production plane
TVector3 y = (beam_cm.Vect().Unit().Cross(-recoil_cm.Vect().Unit())).Unit();
TVector3 x = y.Cross(z);
TVector3 angles( (eta_res.Vect()).Dot(x),
      (eta_res.Vect()).Dot(y),
      (eta_res.Vect()).Dot(z) );
float cosTheta = angles.CosTheta();
```

Decay angular distribution of η(π) encode information of the spin of the ηπ system





Histogramming (taking MM^2 as example)

Header file: DSelector_etapi.h

Define variables and what the class will do

Link to GlueX Style Guidelines

```
TH1F* dHist_mmsq;
```

C file: DSelector_etapi.C

Initialize variables and how the class executes

```
void DSelector_etapi::Init(TTree *locTree)
{
    // Best practices is to include the bin width in the axis labels
    dHist_mmsq = new TH1F("mmsq",";MMsq;Entries / 0.004 GeV",100,-0.2,0.2);
}
Bool_t DSelector_etapi::Process(Long64_t locEntry)
{
    for(UInt_t loc_i = 0; loc_i < Get_NumCombos(); ++loc_i)
    {
        ... apply selections and weights from previous slides... except related to the one you are plotting
        if(bPhotonE*bPhotonTheta*bProtonMomentum*bProton_dEdx*bProtonZ*bChiSq*bUnusedEnergy*bBeamEnergy*bmandelstamt*bMpi0p*bLowMassAltCombo){
            dHist_mmsq->Fill(locMissingMassSquared,weight);}
    } // end of combo loop
}
```

Load histograms **AFTER** running DSelector + draw selections + save as PDF

```
ifarm1801.jlab.org> root -l -b -q drawCuts.C
```

Outputting amptools-ready trees

```
void DSelector_etapi::Init(TTree *locTree)
{
    // Creating a flat tree
    dFlatTreeFileName = "output_flat.root"; //output flat tree (one combo per tree entry), "" for none
    dFlatTreeName = "kin"; //if blank, default name will be chosen

    // Creating new branches in the flat tree
    SetupAmpTools_FlatTree(); // sets most of the branches necesary for AmpTools PWA
```

Include code to create extra branches required by amptools...

```
}
Bool_t DSelector_etapi::Process(Long64_t locEntry)
{
    for(UInt_t loc_i = 0; loc_i < Get_NumCombos(); ++loc_i)
    {
        if(!selection)
        {
                dComboWrapper->Set_IsComboCut(true);
                continue;
        }
        // Filling the branches of the flat tree
```

Include code to fill extra branches required by amptools...

```
        FillAmpTools_FlatTree(locBeamP4, locFinalStateP4);
        Fill_FlatTree(); //for the active combo
    } // end of combo loop
}
```

Basically, 3 steps for amptools tree creation

1. Create flat tree
2. Add some branches
3. Fill the branches

Running the DSelector

File: runDSelector.C

```
void runDSelector()
{
	int proof_Nthreads = 26; // number of threads to process the DSelector with
	gROOT->ProcessLine(".x $(ROOT_ANALYSIS_HOME)/scripts/Load_DSelector.C");

	TString nameOfTree = "pi0eta__B4_M17_M7_Tree";
	//TString nameOfTree = "pi0eta__B4_M7_M17_Tree";
	TChain *chain = new TChain(nameOfTree);

	///// Load all the trees you want to run over
	// - MC simulated flat in mass to determine the acceptance
	//chain->Add("/work/halld/lng/tSlopeSampler/2017_130M/tree_builder/merged/tree_pi0eta__B4_M17_M7_merged*");
	// - Data from the spring 2017 run period
	chain->Add("/cache/halld/RunPeriod-2017-01/analysis/ver52/tree_pi0eta__B4_M17_M7/merged/tree_pi0eta__B4_M17_M7_03*");

	string options = "";
	string outputHistFileName="output_hists.root"; // name of the output root file that contains the histograms
	string outputTreeFileName="output_tree.root"; // name of the output root file that contains the tree

	//// Choice 1: run with proof with your desired number of threads
	DPROOFLiteManager::Process_Chain(chain, "DSelector_etapi.C+", proof_Nthreads, outputHistFileName, outputTreeFileName, options);
	//// Choice 2: run interactively - useful for debugging sometimes when running over small (sub)samples
	//chain->Process("DSelector_etapi.C+", options.data());

	return;
}
```

Wildcard * works
Can use multiple Add lines

Dont forget to change the name!

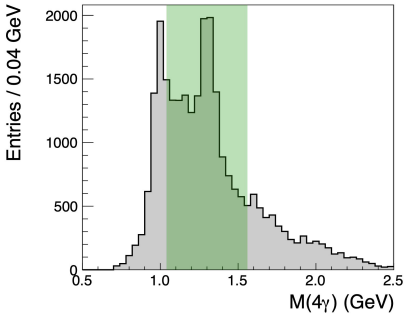
Name of your DSelector
Include “+” for recompiling if some code changed

FLAT MC

SIGNAL/ DATA

```
ifarm1801.jlab.org> root -l -b -q runDSelector.C
```

Final Steps



Additional selection focusing on the $a_2(1320)$ for tutorial →

Run ~**SAME** DSelector over data/MC trees

Make a choice on weighting scheme + runDSelector

```
// AMPTOOLS REQUIRES 4 TREES (data, bkgnd, accmc, genmc)
//    data = selected trees of DATA where signal region has been selected, all weights = 1
//    bkgnd = selected trees of DATA where sidebands have been selected, all weights = -weight
//    accmc = selected trees of ACCEPTANCE MC where signal+sidebands have been selected, all weights = weight
//    genmc = thrown trees created during simulation process
bool signalRegion;
float branchWeight;
//---------CHOICE 1 FOR "data" RUN OVER SIGNAL/DATA-------------
//signalRegion=(pi0_sbweight==1)*(eta_sbweight==1)*(locHistAccidWeightFactor==1); // Keep combos ONLY in the signal region
//branchWeight=1;
//---------CHOICE 2 FOR "bkgnd" RUN OVER SIGNAL/DATA-------------
//signalRegion=!((pi0_sbweight==1)*(eta_sbweight==1)*(locHistAccidWeightFactor==1)); // Keep combos ONLY in the sideband region
//branchWeight=-weight;
//---------CHOICE 3 FOR "accmc" RUN OVER FLAT MC-------------
signalRegion=true; // Keep combos that exist in the signal AND sideband region
branchWeight=weight;
//----------------------
```

OUTPUTS = 3 trees (data,bkgnd,accmc) ready for amplitude analysis

If {data,bkgnd} contains different polarizations - split into different files for amplitude analysis

```
ifarm1801.jlab.org> root -l -b -q 'split_flat_pols.C("D2017_data.root")'
ifarm1801.jlab.org> root -l -b -q 'split_flat_pols.C("D2017_bkgnd.root")
```

Extra

Making a DSelector

```
ifarm1901.jlab.org> MakeDSelector -h

Makes a custom DSelector for the input TTree created by the DANA ANALYSIS library.
1st argument: The input ROOT TTree file name.
2nd argument: The name of the TTree in the input ROOT file that you want to make a selector for.
3rd argument: The base-name of the selector class & files you want to generate.
The generated files will be named "DSelector_<base-name>.C" and "DSelector_<base-name>.h".
```

Filling the output trees

```
for(UInt_t loc_i = 0; loc_i < Get_NumCombos(); ++loc_i)
{
```

... Somewhere here lives basically all the code from previous slides ...

```
//// We can finally multiply all of our selections together to define our final selection criteria.
//      Since we have defined SELECTIONS we have to flip the boolean to get a CUT since the
//      FLAG used asks if the combo should be cut
bool selection=bPhotonE*bPhotonTheta*bProtonMomentum*bProton_dEdx*bProtonZ*bChiSq*bUnusedEnergy*bMMsq*bBeamEnergy*
                bmandelstamt*bMpi0p*bLowMassAltCombo;
```

```
if(!selection)
{
        dComboWrapper->Set_IsComboCut(true);
        continue;
}
```

```
} // end of combo loop
```

```
Bool_t locIsEventCut = true;
for(UInt_t loc_i = 0; loc_i < Get_NumCombos(); ++loc_i) {
        //Set branch array indices for combo and all combo particles
        dComboWrapper->Set_ComboIndex(loc_i);
        // Is used to indicate when combos have been cut
        if(dComboWrapper->Get_IsComboCut())
                continue;
        locIsEventCut = false; // At least one combo succeeded
        break;
}
if(!locIsEventCut && dOutputTreeFileName != "")
        Fill_OutputTree();

return kTRUE;
```

The process function will run over an event and loop over all combos in it

For each combo:

- Define variables / fill histograms
- Determine if combo is cut

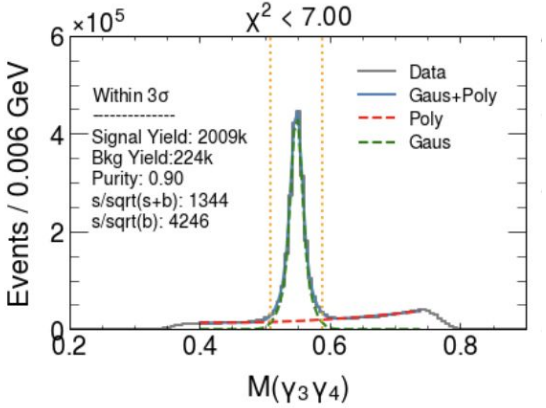
If any combo survives - save the **entire** event

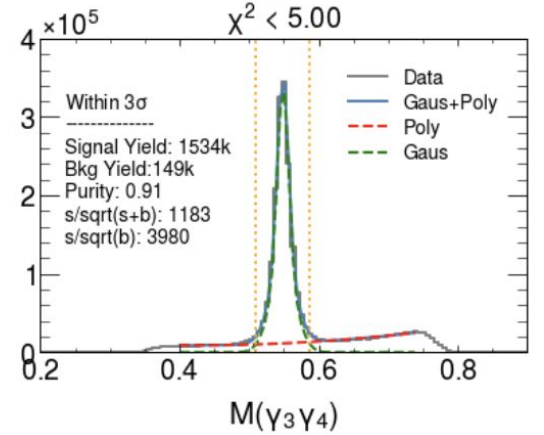
Cut Tuning

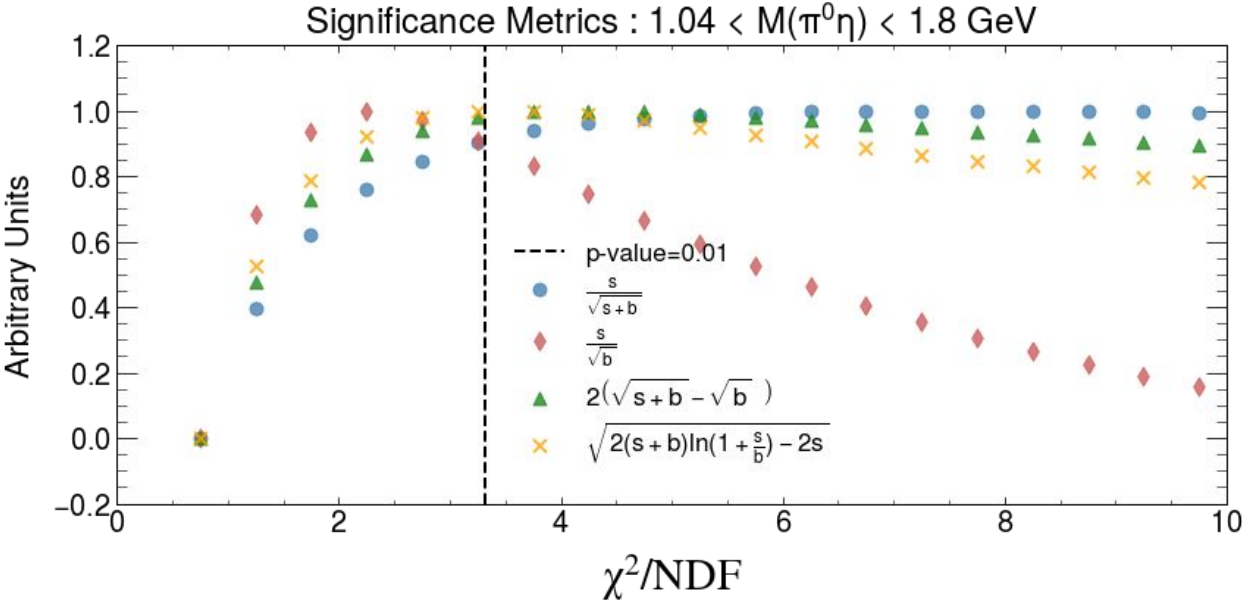
The choice of a selection is not always obvious, i.e. the χ^2 figure of merit

How to choose the selection to make?

1. Requires some distribution with separation power between signal/background, i.e. M(2γ) in this channel
2. For threshold in thresholdOptions
    a. it distribution to extract signal / background yields and construct significance metrics
3. Choose selection that maximizes significance


×10^5
χ² < 7.00
Within 3σ
Signal Yield: 2009k
Bkg Yield:224k
Purity: 0.90
s/sqrt(s+b): 1344
s/sqrt(b): 4246
Data
Gaus+Poly
Poly
Gaus
Events / 0.006 GeV
M(γ3γ4)


×10^5
χ² < 5.00
Within 3σ
Signal Yield: 1534k
Bkg Yield:149k
Purity: 0.91
s/sqrt(s+b): 1183
s/sqrt(b): 3980
Data
Gaus+Poly
Poly
Gaus
M(γ3γ4)


Significance Metrics : 1.04 < M(π⁰η) < 1.8 GeV
Arbitrary Units
p-value=0.01
s/√(s+b)
s/√b
2(√(s+b) − √b)
√(2(s+b)ln(1+s/b) − 2s)
χ²/NDF