Event Selection using FSRoot

Malte Albrecht
Indiana University

GlueX Analysis Workshop 2022

02 / 24 / 2022

What is FSRoot and Why use it?

- Alternative approach to DSelector for event selection
- Based on analysis of “flat” root trees
  - Format is lighter than analysis trees,
  - Facilitates interaction with data in shorter time
- Each *combo* that survives is a new entry to the tree
- Get FSRoot here: https://github.com/remitche66/FSRoot.git
- Documentation available in repository

Notes on the FSRoot Package

Ryan Mitchell

May 11, 2022 (v4.0)

Abstract

FSRoot is a set of utilities, built around the CERN ROOT framework, that can be used to analyze a variety of final states (FS) produced in particle physics experiments. This document provides a short introduction to FSRoot.

Contents

Example: $\gamma p \to \eta\pi^0 p$

- Focus on $\gamma p \to \eta\pi^0 p$ for this tutorial
- Using analysis trees produced by ReactionFilter:
    - **pi0eta__B4_M7_M17_Tree**
    - 4 beam bunches
    - Pi0 and eta masses not constrained in kinematic fit
    - Mass windows for two-photon combinations applied
    - Can contain multiple combos per event (different beam photons, swapping of final state photons between eta / pi0 with both in mass windows)

Step 1: FlattenForFSRoot

- Helper program to create flat trees: FlattenForFSRoot in hd_utilities repository
- Reduces size of trees, possibility to apply some pre-selection cuts:

```
Usage:
  flatten  -in    <input file name>                      (required)
           -out   [output file name or none]             (default: none)
                   (if none, just print info and quit)
           -mc    [is this mc? -1, 0, or 1]              (default: -1)
                   (-1: determine automatically; 0: no; 1: yes)
           -mctag [MCExtras_MCDecayCode2_MCDecayCode1]   (default: none)
                   (pick out a single final state from MC)
           -chi2  [optional Chi2/DOF cut value]          (default: 1000)
           -shQuality  [optional shower quality cut value] (default: -1 (no cut))
           -massWindows  [pi0, eta, (A)Lambda, Ks windows (GeV)] (default: -1 (no cut))
                        (uses the most constrained four-momenta)
           -numUnusedTracks   [optional cut (<= cut)]    (default: -1 (no cut))
           -numUnusedNeutrals [optional cut (<= cut)]    (default: -1 (no cut))
           -numNeutralHypos   [optional cut (<= cut)]    (default: -1 (no cut))
           -usePolarization   [get polarization angle from RCDB? 0 or 1]   (default: 0)
           -addPID    [include PID info in the output tree? 0 or 1]   (default: 0)
           -mcChecks  [check for baryon number violation, etc.,
                       when parsing truth information?  0 or 1] (default: 1)
           -safe  [check array sizes?  0 or 1]           (default: 1)
           -print [print to screen:
                   -1 (less); 0 (regular); 1 (more)]     (default: 0)
```

- Example for eta pi0 run (note: wildcards allowed):

```
$HD_UTILITIES_HOME/FlattenForFSRoot/flatten -in <path_to_trees>/tree_pi0eta__B4_M17_M7_031057.root
      -out tree_pi0eta__B4_M17_M7_FLAT_031057.root
      -chi2 15 -massWindows 0.3 -numNeutralHypos 6 -numUnusedTracks 1 -usePolarization 1
```

- Size of analysis trees: **~1010Gb**   Size of flattened trees with given cuts: **~20Gb**

Step 2: Basic Cuts and Skimming Trees

- Detailed description how to use cuts in documentation, focus on example for etapi0 in a2 mass region here

- Cuts are chosen to be same as for DSelector analysis:

```
// DEFINITION OF CUTS:
  // STATIC CUTS
  FSCut::defineCut("unusedE","EnUnusedSh<0.1");                              // UnusedEnergy < 0.1GeV
  FSCut::defineCut("unusedTracks","NumUnusedTracks<1");                      // No unsused tracks
  FSCut::defineCut("zProton","ProdVz>=52&&ProdVz<=78");                      // Production vertex z-position
  FSCut::defineCut("protMom","MOMENTUM([p+])>=0.3");                         // Proton momentum > 0.3GeV/c
  FSCut::defineCut("cet0103","OR(abs(-1*MASS2(GLUEXTARGET,-[p+])-0.2)<0.1)"); // 0.1 < t < 0.3
  FSCut::defineCut("e8288","(EnPB>8.2&&EnPB<8.8)");                          // 8.2 < E_beam < 8.8
  FSCut::defineCut("chi2","Chi2DOF<3.3");                                    // Chi2/ndf < 3.3
  FSCut::defineCut("photFiducialA","(acos(COSINE([eta]a))*180/3.141>2.5 &&
                                    acos(COSINE([eta]a))*180/3.141<10.3) || (acos(COSINE([eta]a))*180/3.141>11.9)");
  // (same fuducial cut for remaining photons)

  FSCut::defineCut("rejectOmega","!((MASS([pi0]a,[eta]a)<0.15 && MASS([pi0]b,[eta]b)<0.15) ||
                                   (MASS([pi0]a,[eta]b)<0.15 && MASS([pi0]b,[eta]a)<0.15) ||
                                   (MASS([pi0]a,[eta]a)<0.12 && MASS([pi0]b,[eta]a)<0.12) ||
                                   (MASS([pi0]a,[eta]b)<0.12 && MASS([pi0]b,[eta]b)<0.12))");

  FSCut::defineCut("a2","MASS([eta],[pi0])>=1.04 && MASS([eta],[pi0])<=1.56");
```

- Use these cuts to skim the trees and further reduce size

- **Before doing that**: Make basic plots for signal channel, flat MC, then look at data, refine/optimize cuts

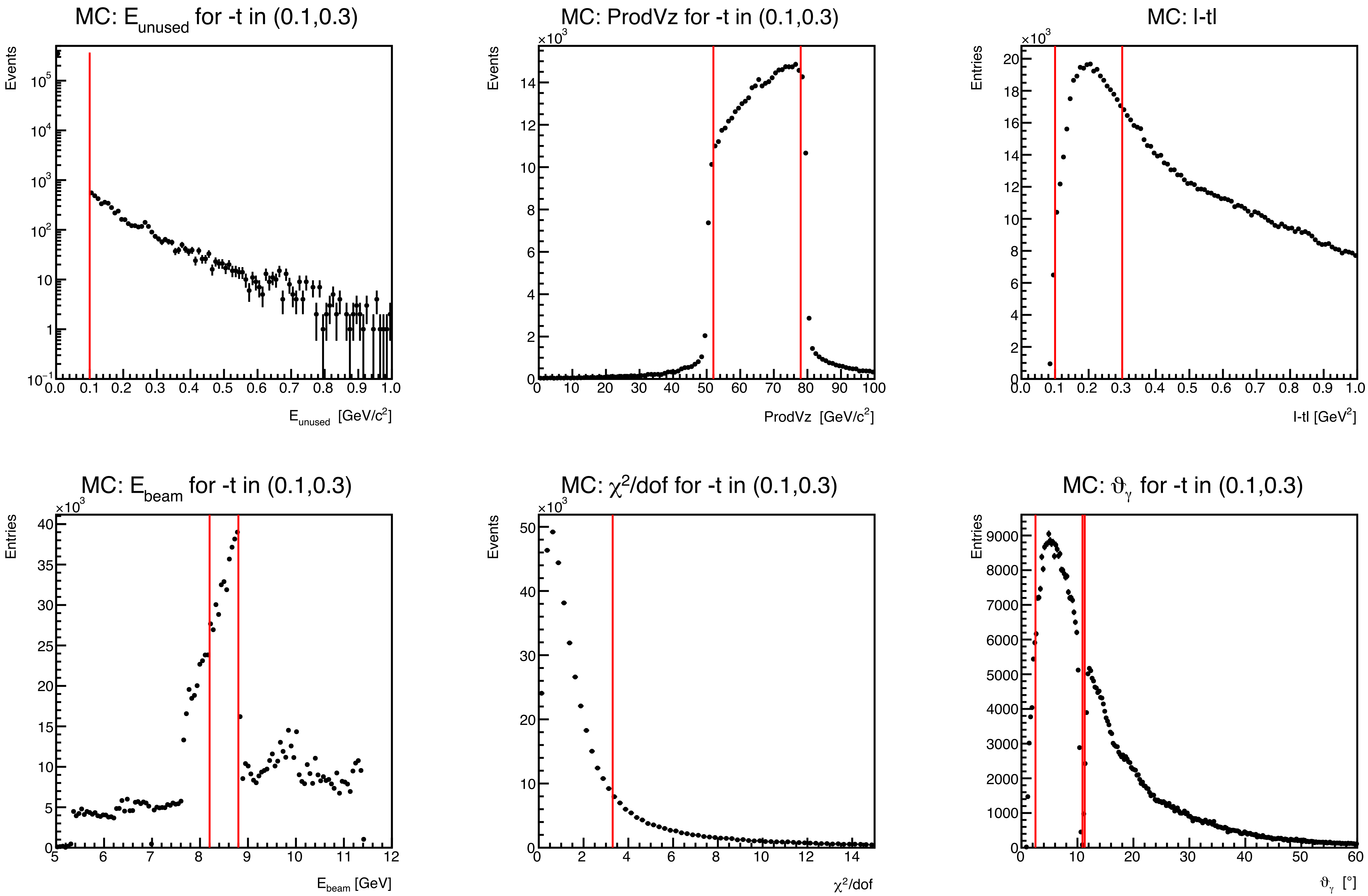
Step 3: Making basic Plots

- Look at MC for your signal channel, in this case $\gamma p \to \eta\pi^0 p$ first
- One example how to plot an invariant mass:

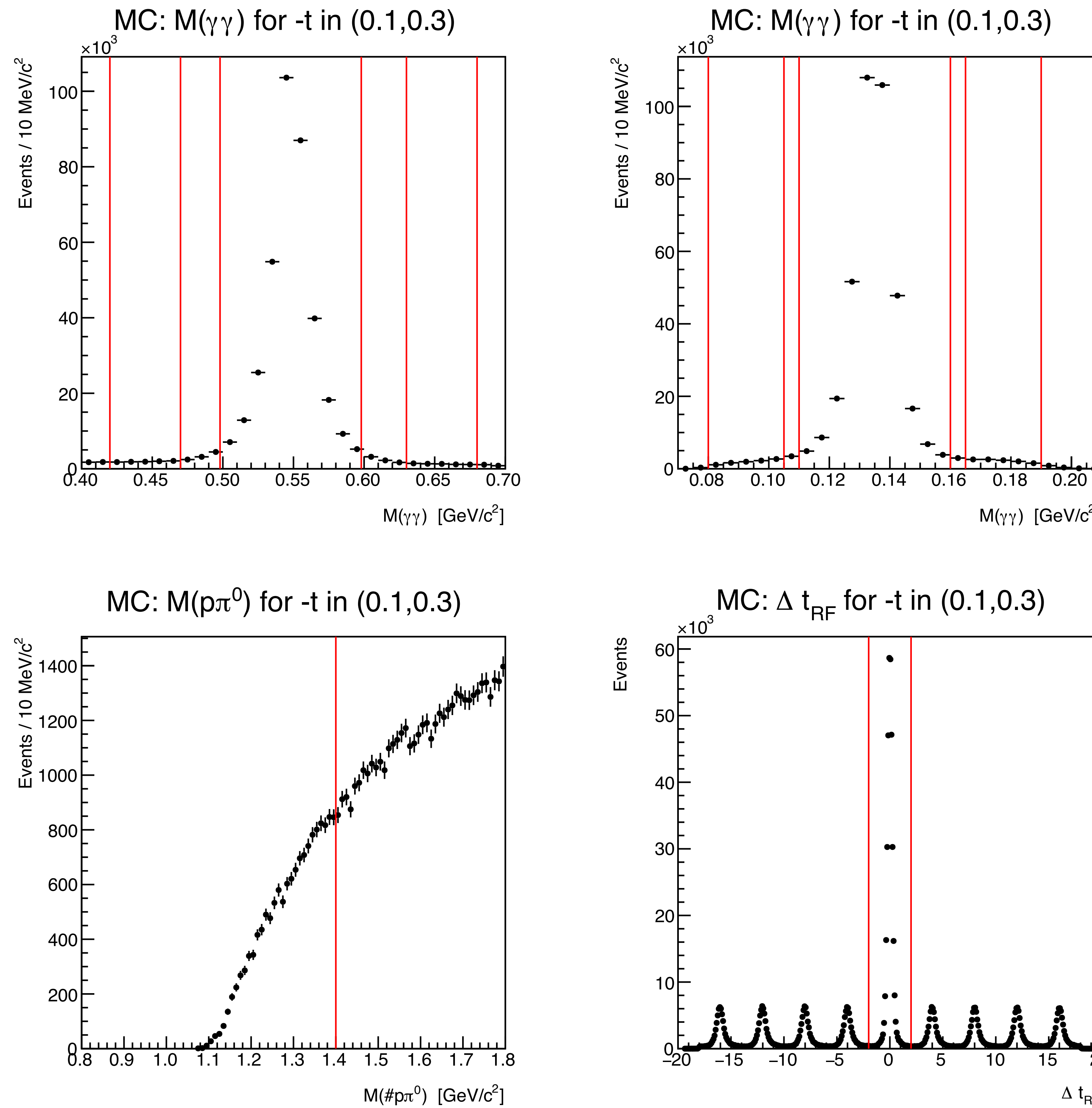
```
TH1F* hM3 = FSModeHistogram::getTH1F(FND,NT,"m101_1","MASS([p+],[pi0])","(100,0.8,1.8)",
            "CUT(unusedTracks,zProton,unusedE,chi2,cet0103,e8288,photFiducialA,photFiducialB,
             photFiducialC,photFiducialD,rf,eta,pi0,rejectOmega,protMom)");
hM3->SetTitle("MC: M(p#pi^{0}) for -t in (0.1,0.3)");
hM3->SetXTitle("M(#p#pi^{0})  [GeV/c^{2}]");
hM3->SetYTitle("Events / 10 MeV/c^{2}");
TLine* cutDelta = new TLine(1.4,0,1.4,hM3->GetMaximum());
cutDelta->SetLineColor(kRed);
hM3->Draw();
cutDelta->Draw("same");
```

- Lots of useful macros defined in FSRoot, that let you plot various variables of interest
- Many examples are in plots.C script

Step 3: Basic Plots using MC



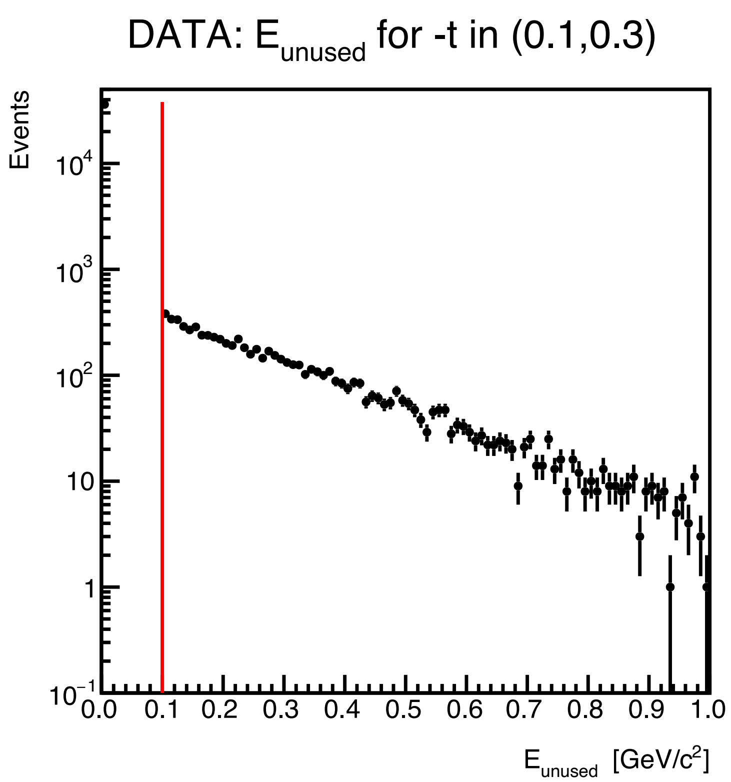
- Verify all cuts are appropriate for pure signal MC first
- No model needed, flat MC is a good place to start

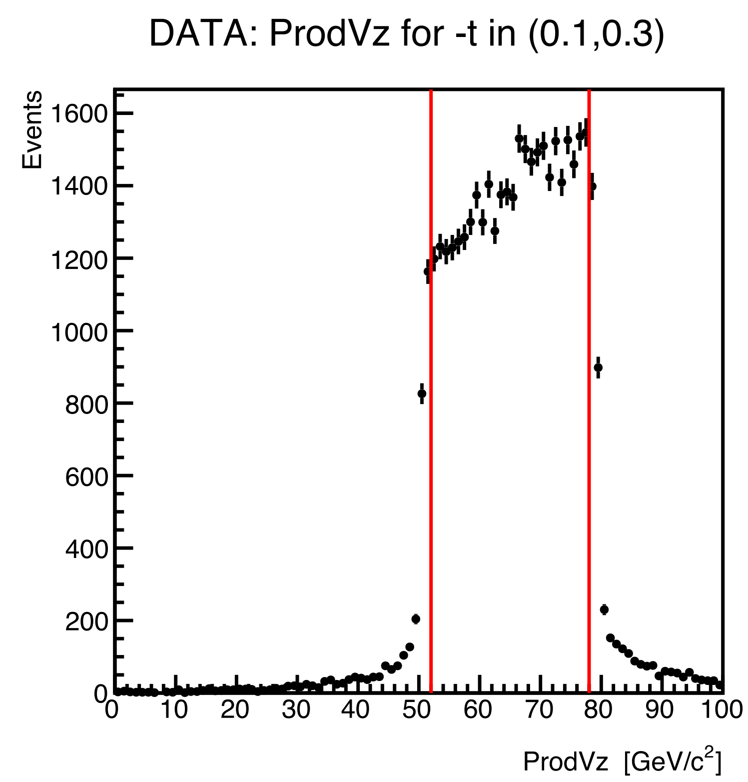


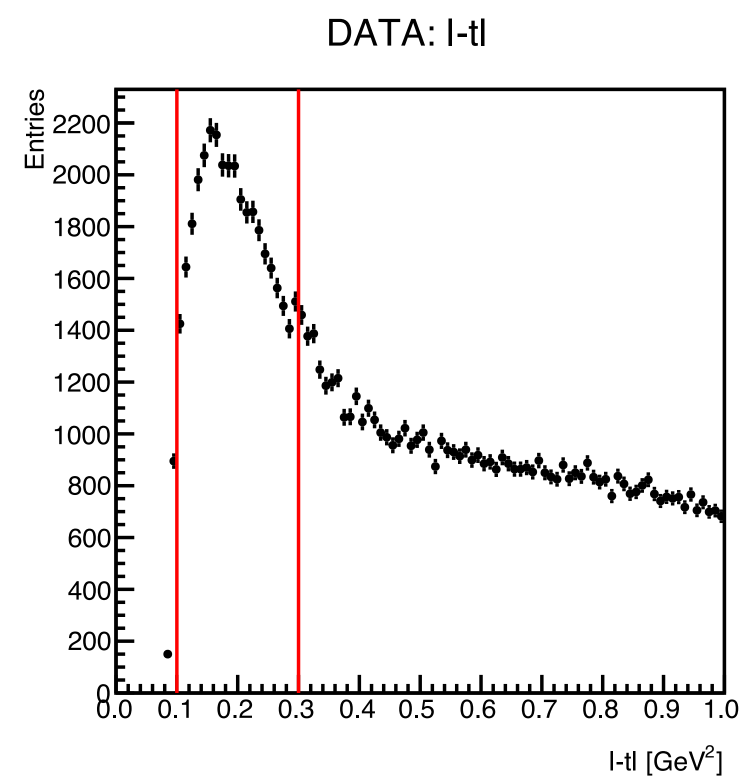
Step 3: Analysing bggen Monte Carlo Data

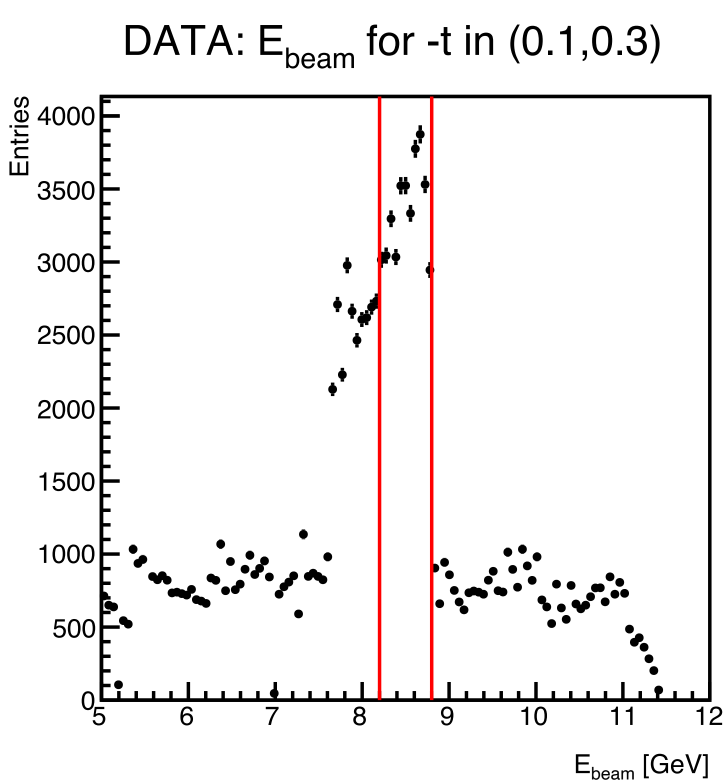
- Two plots for bggen MC here to illustrate bg rejection by omega cut

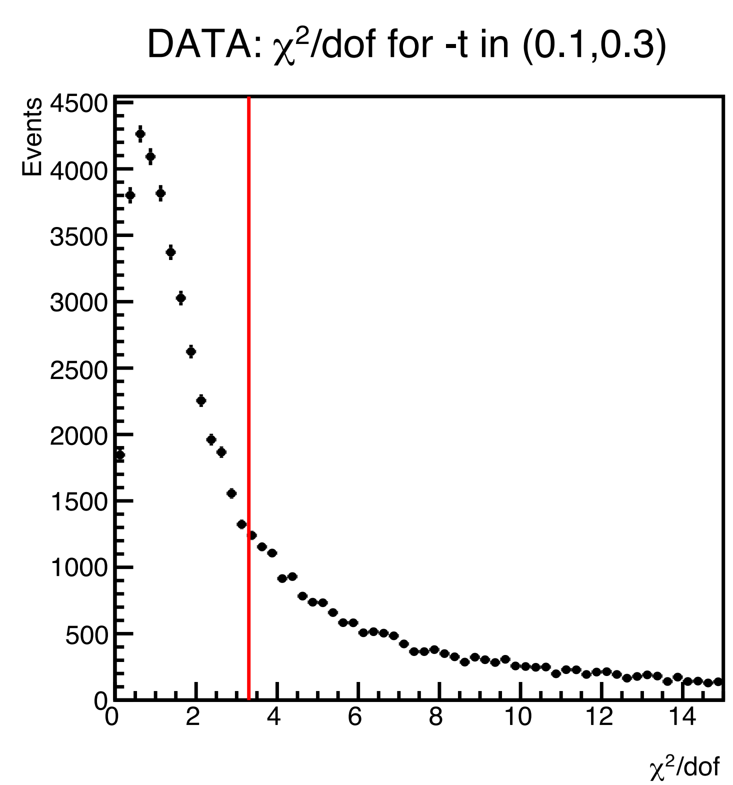
Step 4: Basic Plots for DATA

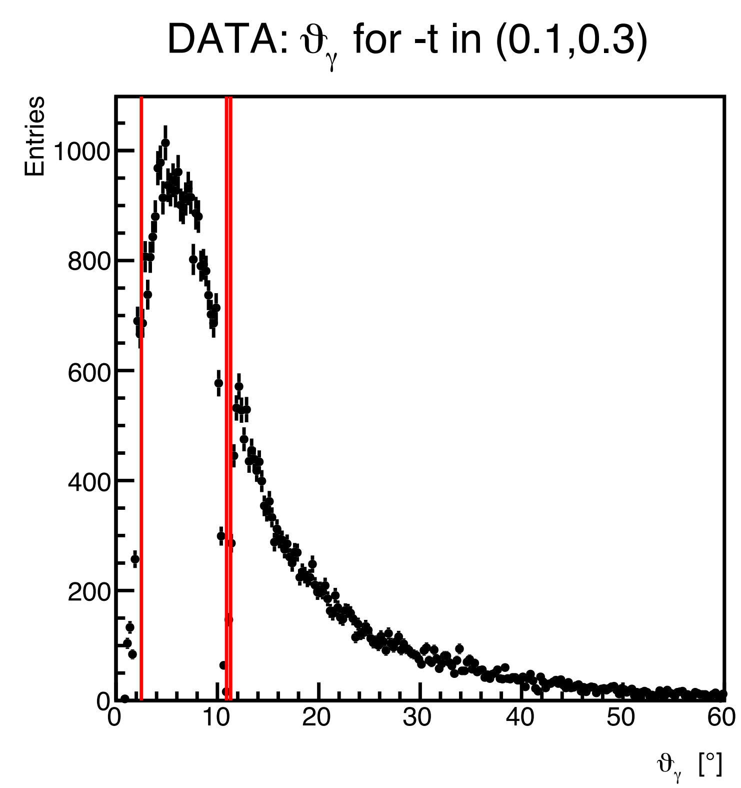






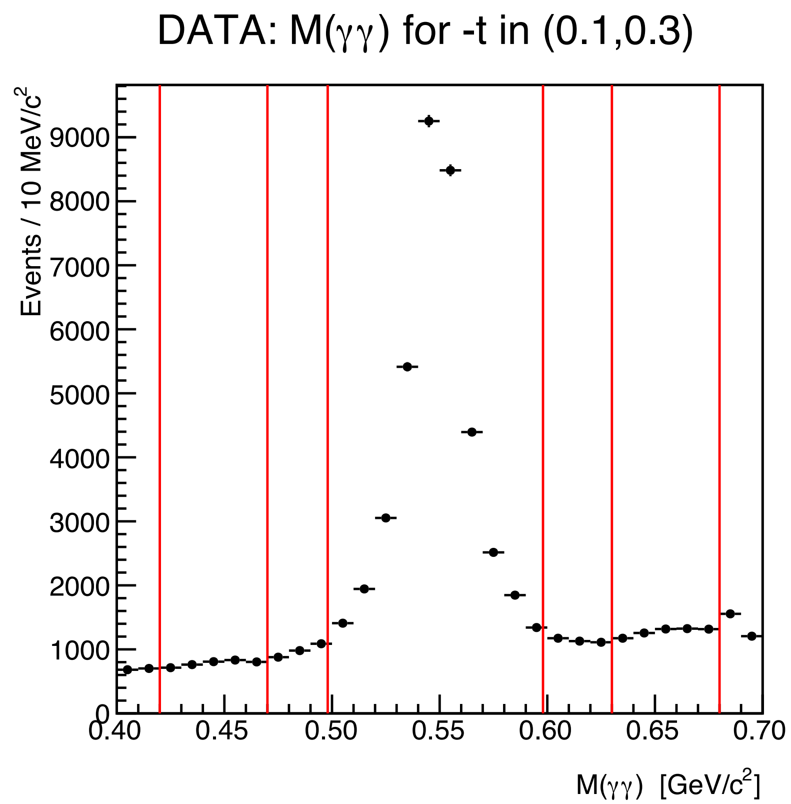


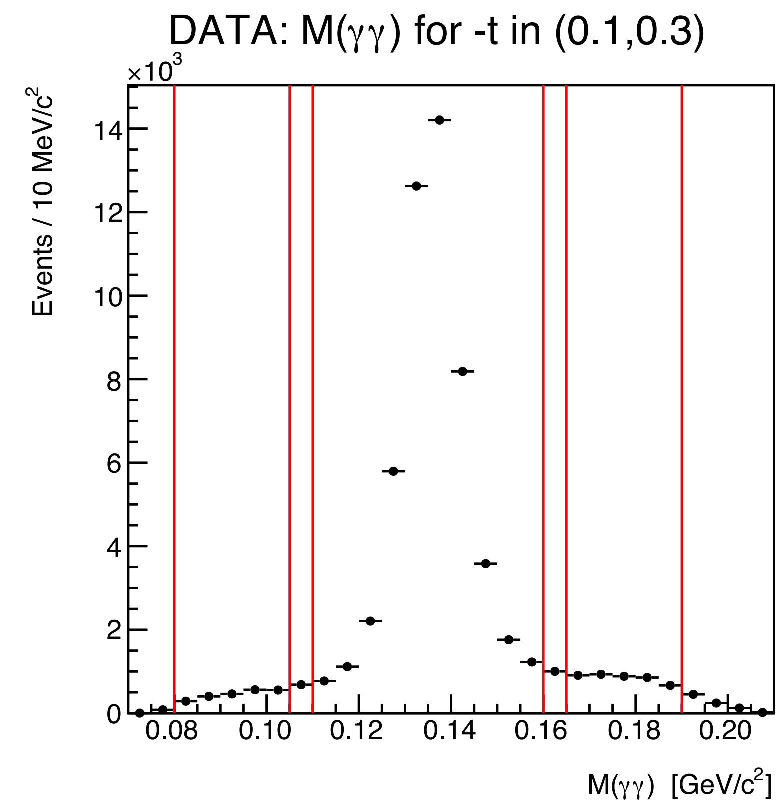


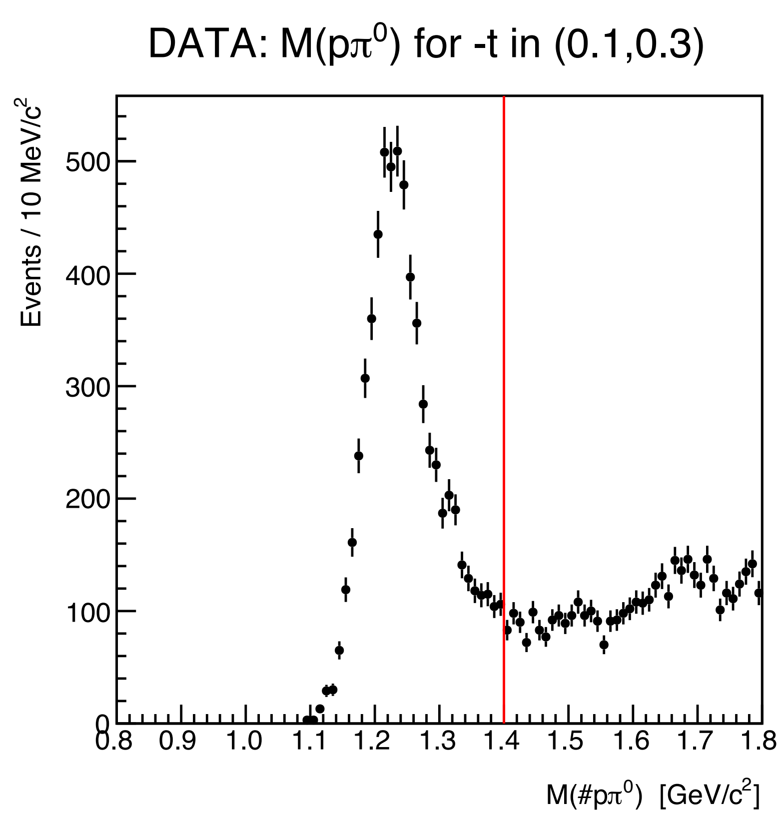


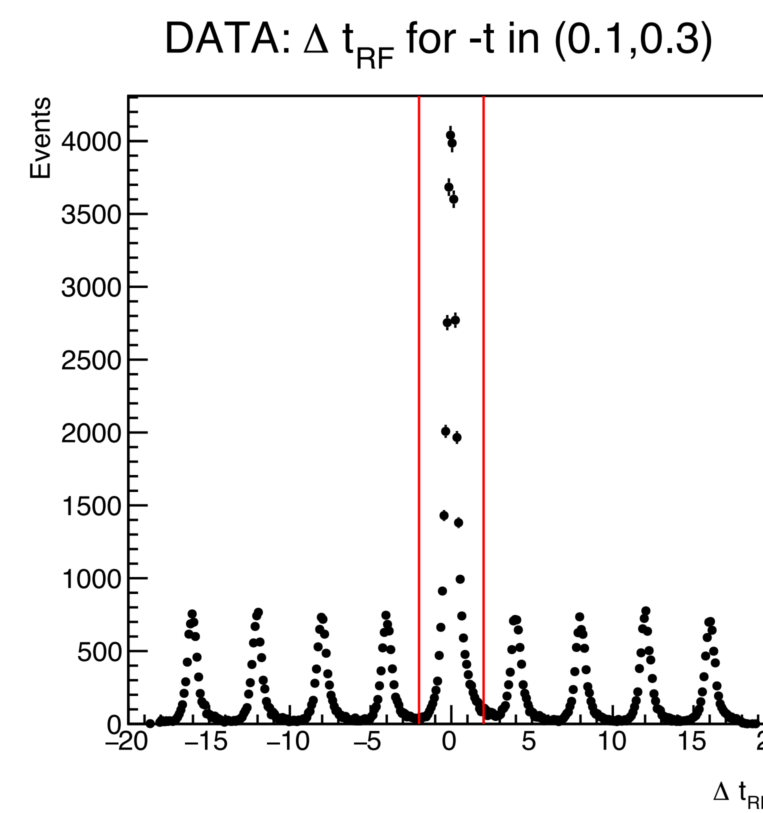
Step 4: Basic Plots for DATA

- Check performance of cuts in data, refine as necessary
- Optimize cuts
- Once you're happy with all selection criteria, produce skimmed trees for PWA

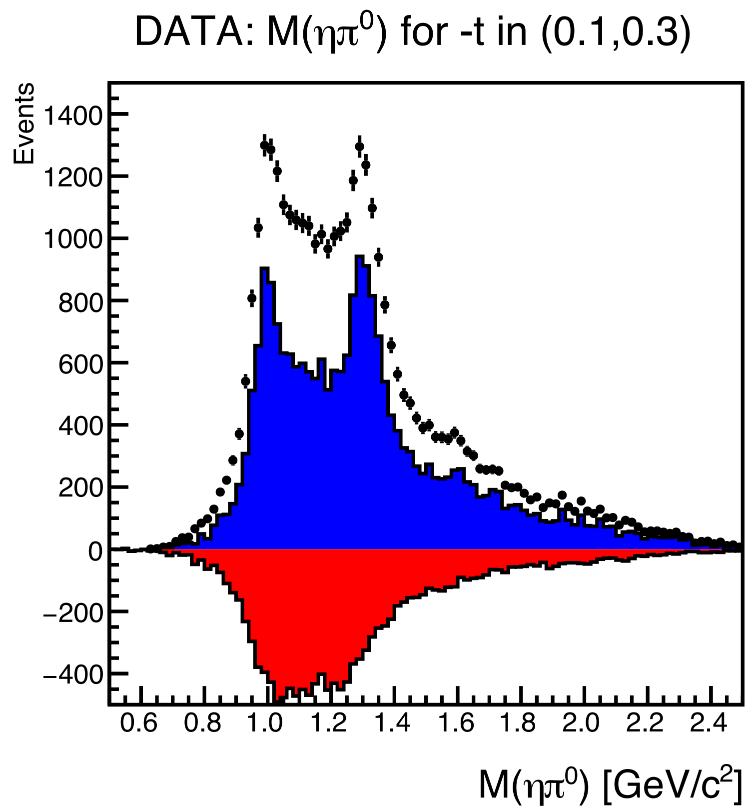


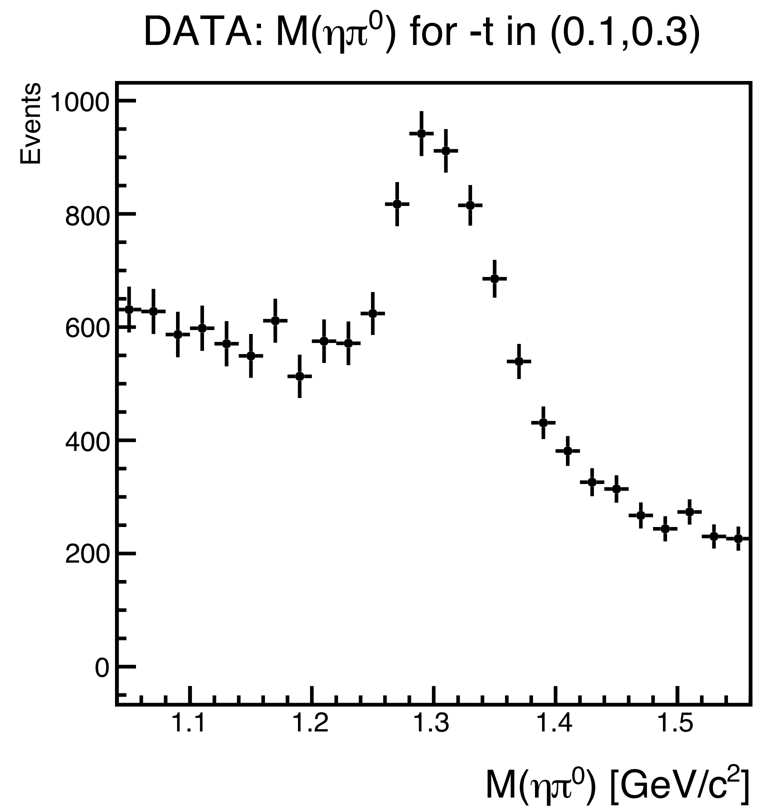


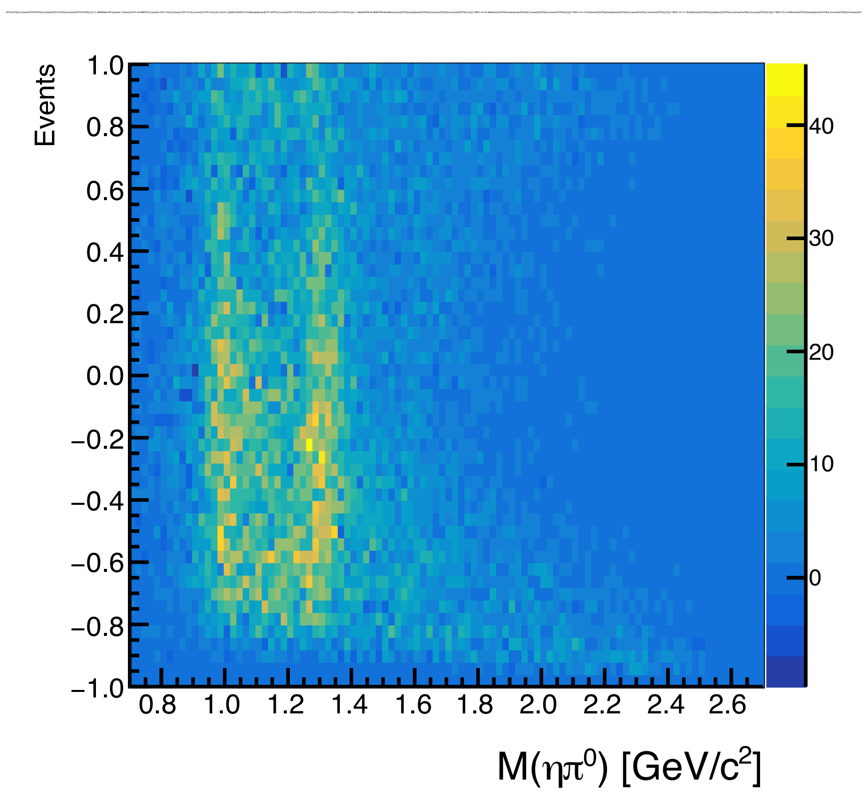




Step 4: Basic Plots for Data







- Accidental and sideband subtractions: Check what will be subtracted
- FSRoot handles multi-dimensional subtractions internally (e.g. accidental + pi0 sideband + eta sideband) by calculating corresponding weights from sidebands defined in CUT definitions:

```
TH1F* hMetapi = FSModeHistogram::getTH1F(FND,NT,"m101_1","MASS([eta],[pi0])","(100,0.5,2.5)",
                "CUT([...list of all cuts...])");
TH1F* hMetapiSig = FSModeHistogram::getTH1F(FND,NT,"m101_1","MASS([eta],[pi0])","(100,0.5,2.5)",
                "CUT([...list of all cuts...])*CUTWT(rf,eta,pi0)");
TH1F* hMetapiBg = FSModeHistogram::getTH1F(FND,NT,"m101_1","MASS([eta],[pi0])","(100,0.5,2.5)",
                "CUT([...list of all cuts...])*CUTSBWT(rf,eta,pi0)*(-1.0)");
```

Step 5: Skimming Trees

- Which trees do we want to produce?
  - To make plots / inspect data / refine cuts:
    All cuts applied that are not used for sideband / background subtraction
  - For PWA / AmpTools:
    - DATA: All Cuts applied, select signal region only, all weights=1
    - BKGND: All Cuts applied, select appropriate sideband regions with corresponding weights
    - ACCMC: Flat signal channel MC, reconstructed and subjected to the same cuts as data, signal region only
    - GENMC: Thrown MC with no cuts applied
- Example for one polarization, DATA and BKGND trees:

```
// Write out skimmed DATA trees with GENERAL CUTS applied:
FSModeTree::skimTree(FND0,NT,"","tree_pi0eta__B4_M17_M7_DATA_sp17_pol0_GENERAL_SKIM_A2.root",
                    "CUT([...list of cuts...])");
// Write out skimmed tree with GENERAL CUTS applied for SIGNAL REGION ONLY:
FSModeTree::skimTree("tree_pi0eta__B4_M17_M7_DATA_sp17_pol45_GENERAL_SKIM_A2.root",NT,"",
                    "tree_pi0eta__B4_M17_M7_DATA_sp17_pol45_SIGNAL_SKIM_A2.root","CUT(rf,eta,pi0)");
// Write out skimmed tree with GENERAL CUTS applied with SIDEBAND WEIGHTS:
FSModeTree::skimTree("tree_pi0eta__B4_M17_M7_DATA_sp17_pol0_GENERAL_SKIM_A2.root",NT,"",
                    "tree_pi0eta__B4_M17_M7_DATA_sp17_pol0_SIDEBANDS_SKIM_A2.root","CUTSBWT(rf,eta,pi0)");
vector< pair<TString,TString> > friendTreeContents;
friendTreeContents.push_back(pair<TString,TString>("weight","CUTSBWT(rf,eta,pi0)"));
FSModeTree::createFriendTree("tree_pi0eta__B4_M17_M7_DATA_sp17_pol0_SIDEBANDS_SKIM_A2.root",NT,"","weight",friendTreeContents);
```

Output: Writing out Trees to use in PWA

•

Summary & Outlook