Introduction to ROOT

an informal tutorial

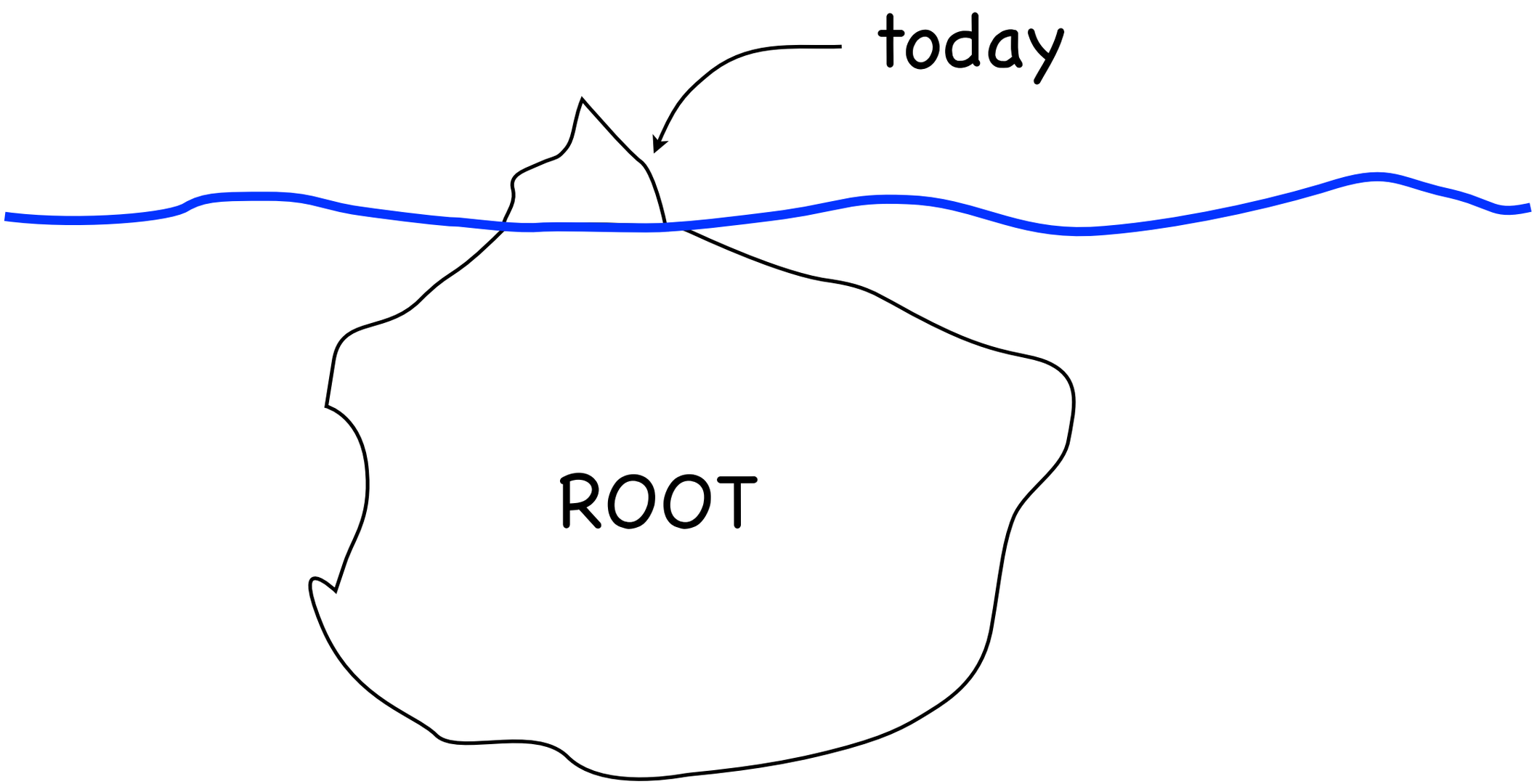
Florian Bonnet

INFN - Sezione di Padova

July 6th 2011

Introduction to ROOT

an informal tutorial



Content

- Turnkey program structures for events studies
- Empirical knowledge gather from experience

- I'm not an expert and don't know everything about ROOT
- Don't try to understand all the subtleties, it is ok to copy-paste something that is working

Why ROOT ?

Events Generators ----> Events files ----> Find/exploit unique kinematics features

Read Events ⟶ { Identify final state particles ; Get their kinematic info.

Calculate physical quantities ⟶ Build fonctions of the kin. info.

Make nice plots ⟶ To identify peculiar behaviors

Apply cuts ⟶ Signal Vs Backgrounds

Events : LHCO format

particle type

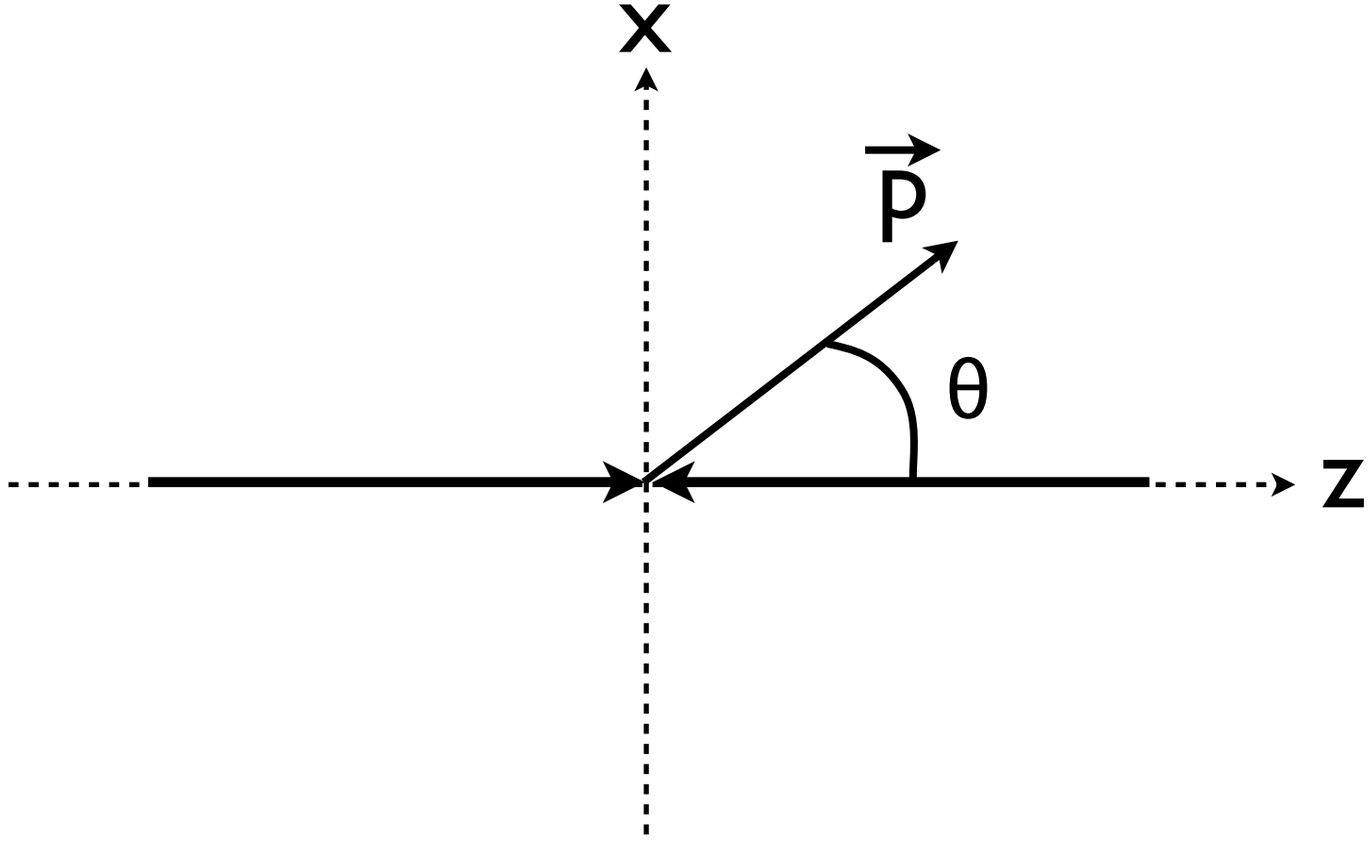
#	typ	eta	phi	pt	jmas	ntrk	btag	had/em	dum1	dum2
1	1	0.034	1.540	14.24	0.00	-1.0	0.0	0.00	0.0	0.0
2	1	0.331	5.858	28.85	0.00	1.0	0.0	0.00	0.0	0.0
3	2	1.724	0.018	21.45	0.11	1.0	0.0	0.05	0.0	0.0
4	2	-0.375	1.591	48.25	0.11	-1.0	0.0	0.05	0.0	0.0
5	6	0.000	3.981	72.19	0.00	0.0	0.0	0.00	0.0	0.0

0 = photon
1 = electron
2 = muon
3 = hadronically-decaying taus
4 = jets
6 = missing energy

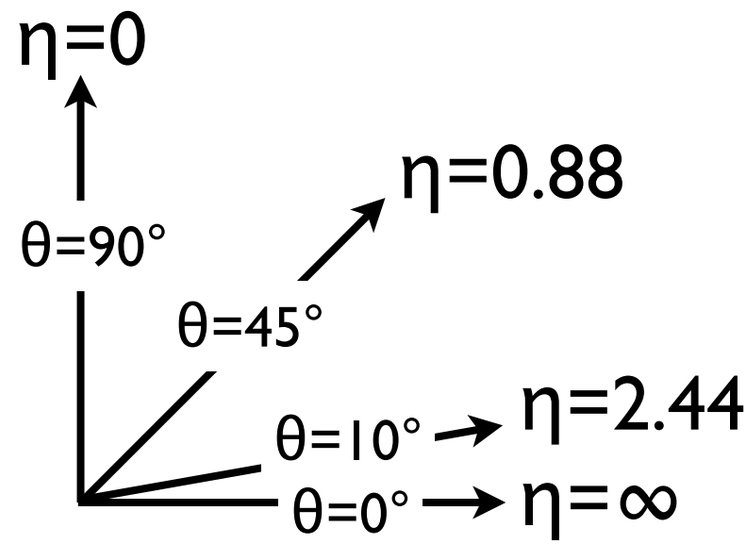
Events : LHCO format

pseudo-rapidity

#	typ	eta	phi	pt	jmas	ntrk	btag	had/em	dum1	dum2
1	1	0.034	1.540	14.24	0.00	-1.0	0.0	0.00	0.0	0.0
2	1	0.331	5.858	28.85	0.00	1.0	0.0	0.00	0.0	0.0
3	2	1.724	0.018	21.45	0.11	1.0	0.0	0.05	0.0	0.0
4	2	-0.375	1.591	48.25	0.11	-1.0	0.0	0.05	0.0	0.0
5	6	0.000	3.981	72.19	0.00	0.0	0.0	0.00	0.0	0.0



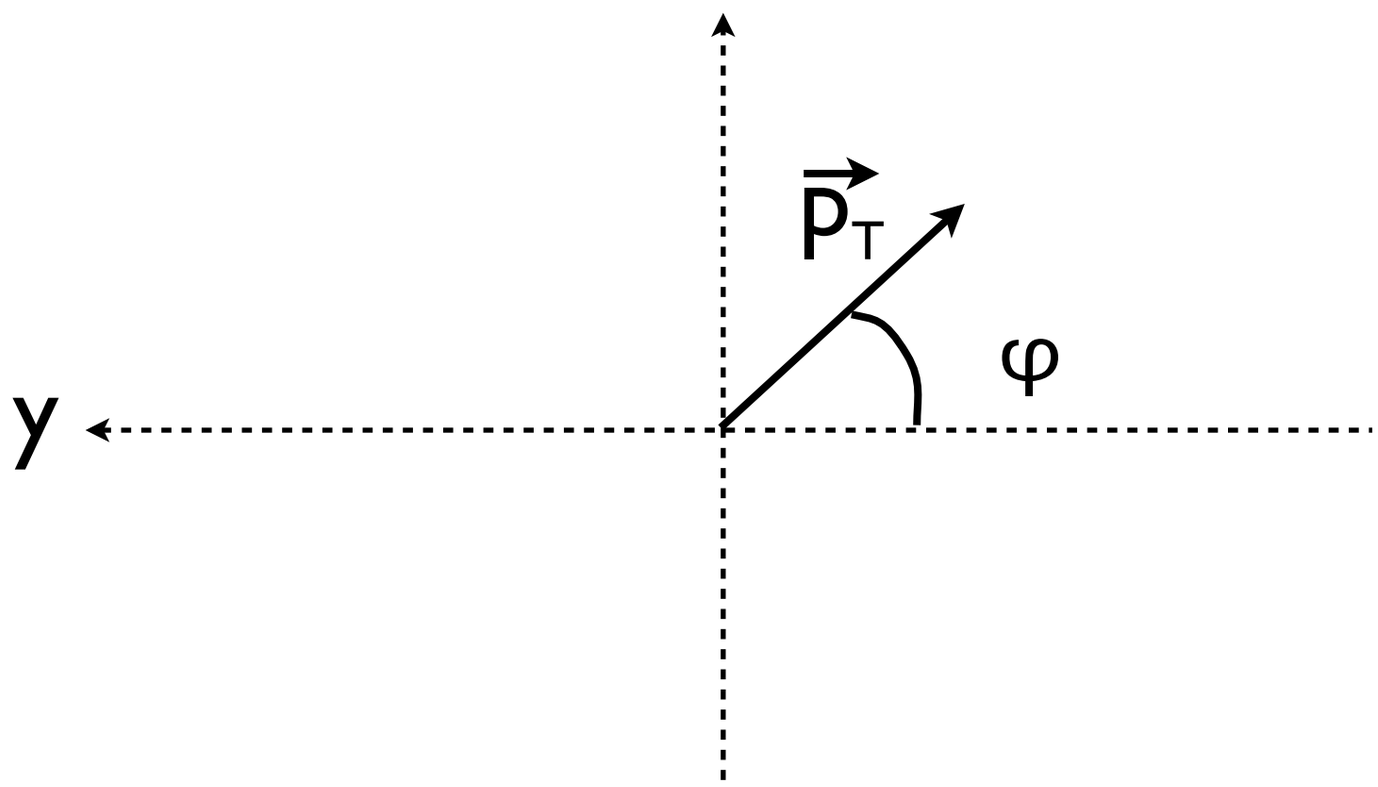
$$\eta = - \ln(\tan(\theta)/2)$$



Events : LHCO format

transverse plane

#	typ	eta	phi	pt	jmas	ntrk	btag	had/em	dum1	dum2
1	1	0.034	1.540	14.24	0.00	-1.0	0.0	0.00	0.0	0.0
2	1	0.331	5.858	28.85	0.00	1.0	0.0	0.00	0.0	0.0
3	2	1.724	0.018	21.45	0.11	1.0	0.0	0.05	0.0	0.0
4	2	-0.375	1.591	48.25	0.11	-1.0	0.0	0.05	0.0	0.0
5	6	0.000	3.981	72.19	0.00	0.0	0.0	0.00	0.0	0.0



Events : LHCO format

invariant mass

#	typ	eta	phi	pt	jmas	ntrk	btag	had/em	dum1	dum2
1	1	0.034	1.540	14.24	0.00	-1.0	0.0	0.00	0.0	0.0
2	1	0.331	5.858	28.85	0.00	1.0	0.0	0.00	0.0	0.0
3	2	1.724	0.018	21.45	0.11	1.0	0.0	0.05	0.0	0.0
4	2	-0.375	1.591	48.25	0.11	-1.0	0.0	0.05	0.0	0.0
5	6	0.000	3.981	72.19	0.00	0.0	0.0	0.00	0.0	0.0

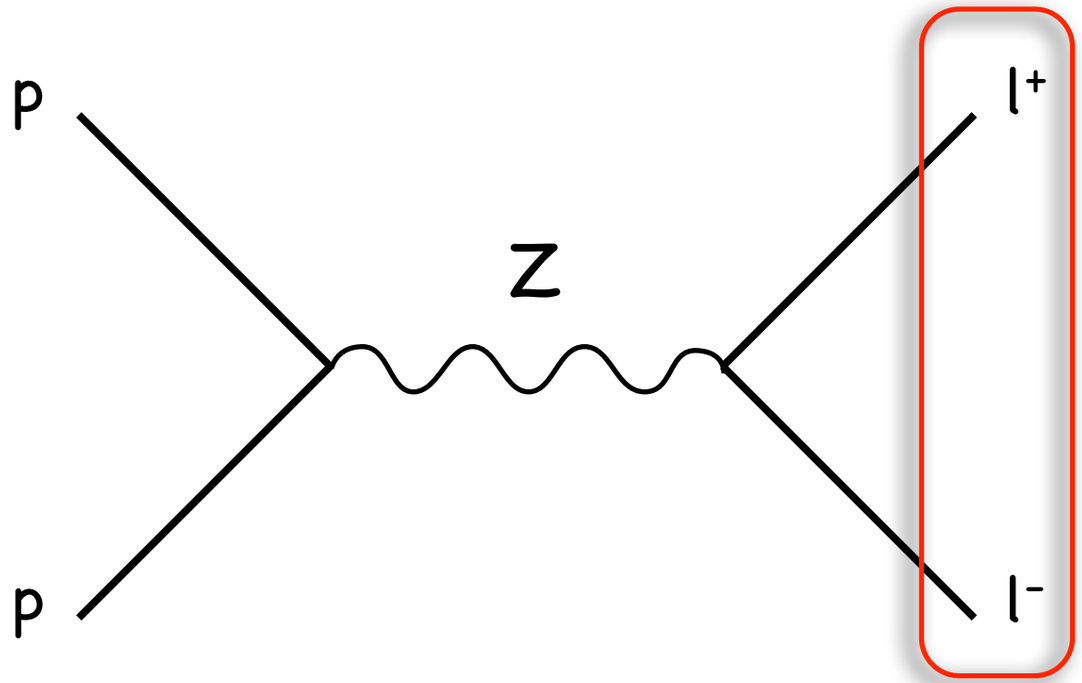
Events : LHCO format

charge for leptons

#	typ	eta	phi	pt	jmas	ntrk	btag	had/em	dum1	dum2
1	1	0.034	1.540	14.24	0.00	-1.0	0.0	0.00	0.0	0.0
2	1	0.331	5.858	28.85	0.00	1.0	0.0	0.00	0.0	0.0
3	2	1.724	0.018	21.45	0.11	1.0	0.0	0.05	0.0	0.0
4	2	-0.375	1.591	48.25	0.11	-1.0	0.0	0.05	0.0	0.0
5	6	0.000	3.981	72.19	0.00	0.0	0.0	0.00	0.0	0.0

1st example :
Reconstructing M_Z

Reconstructing M_Z



invariant mass should show a pic in M_Z

Let's see from
Zmass_pgs_events.lhco

Open Zmass.C

Reconstructing M_Z

- Open the event file

```
const int nSims = 1;  char * fileNames[nSims] = {"Zmass_pgs_events.lhco"};
```

Reconstructing M_Z

- Open the event file

```
const int nSims = 1;  char * fileNames[nSims] = {"Zmass_pgs_events.lhco"};
```

- Prepare the analysis
  - Get ready to store all final particles of an event

```
struct particle {
  int type;
  int index;
  double eta;
  double phi;
  double pt;
  double jma;
  double ntracks;
  double btag;
  double hadem;
  double dummy1;
  double dummy2;
};
```

#	typ	eta	phi	pt	jmas	ntrk	btag	had/em	dum1	dum2
1	1	0.034	1.540	14.24	0.00	-1.0	0.0	0.00	0.0	0.0
2	1	0.331	5.858	28.85	0.00	1.0	0.0	0.00	0.0	0.0
3	2	1.724	0.018	21.45	0.11	1.0	0.0	0.05	0.0	0.0
4	2	-0.375	1.591	48.25	0.11	-1.0	0.0	0.05	0.0	0.0
5	6	0.000	3.981	72.19	0.00	0.0	0.0	0.00	0.0	0.0

Reconstructing M_Z

- Open the event file

```
const int nSims = 1;  char * fileNames[nSims] = {"Zmass_pgs_events.lhco"};
```

- Prepare the analysis
  - Get ready to store all final particles of an event

```
struct particle {
  int type;
  int index;
  double eta;
  double phi;
  double pt;
  double jma;
  double ntracks;
  double btag;
  double hadem;
  double dummy1;
  double dummy2;
};
```

```
vector<particle> photons;
vector<particle> leptons;
vector<particle> electrons;
vector<particle> muons;
vector<particle> hadtaus;
vector<particle> jets;
vector<particle> bjets;
vector<particle> MET;
```

Reconstructing M_Z

- Open the event file

```
const int nSims = 1;  char * fileNames[nSims] = {"Zmass_pgs_events.lhco"};
```

- Prepare the analysis
  - Get ready to store all final particles of an event
  - Intermediate variables
  - Lorentz vector corresponding to final state particles

```
TLorentzVector lepton1;
TLorentzVector lepton2;
```

→ (P_x, P_y, P_z, E) + Several Built-in functions

Reconstructing M_Z

- Open the event file

```
const int nSims = 1;  char * fileNames[nSims] = {"Zmass_pgs_events.lhco"};
```

- Prepare the analysis
  - Get ready to store all final particles of an event
  - Intermediate variables
  - Lorentz vector corresponding to final state particles
  - Define the histograms you will plot

```
TH1F * Histo_mll = new TH1F("Histo_mll","mll",nBins_mll,60,120);
```

Histogram with 1 float/channel

Reconstructing M_Z

- Open the event file

```
const int nSims = 1;  char * fileNames[nSims] = {"Zmass_pgs_events.lhco"};
```

- Prepare the analysis
  - Get ready to store all final particles of an event
  - Intermediate variables
  - Lorentz vector corresponding to final state particles
  - Define the histograms you will plot

```
TH1F * Histo_mll = new TH1F("Histo_mll","mll",nBins_mll,60,120);
```

Name Title #bins range

Reconstructing M_Z

- Open the event file

```
const int nSims = 1;  char * fileNames[nSims] = {"Zmass_pgs_events.lhco"};
```

- Prepare the analysis
  - Get ready to store all final particles of an event
  - Intermediate variables
  - Lorentz vector corresponding to final state particles
  - Define the histograms you will plot

- Main Program :
  - Same name as the file
  - Read all events and for each one store particles
  - ...

Reconstructing M_Z

- Pre-selection :
  - You want only 2 leptons

```
nl = leptons.size();
if (nl == 2){
}
```

- Using TLorentzVector

```
lepton1.SetPtEtaPhiM(leptons[0].pt,leptons[0].eta,leptons[0].phi,leptons[0].jma);
```

→ Built-in function that defines (P_x, P_y, P_z, E) from (P_T, θ, φ, m)

Reconstructing M_Z

- Pre-selection :
  - You want only 2 leptons

```
nl = leptons.size();
if (nl == 2){
}
```

- Using TLorentzVector

```
lepton1.SetPtEtaPhiM(leptons[0].pt,leptons[0].eta,leptons[0].phi,leptons[0].jma);
```

→ Built-in function that defines (P_x, P_y, P_z, E) from (P_T, θ, φ, m)

- Calculate invariant mass

```
mll = (lepton1+lepton2).M();
```

→ Built-in function that calculate invariant mass

- Fill the histogram

```
Histo_mll->Fill(mll);
```

Reconstructing M_Z

- Plotting
  - Canvas = windows where histograms appear

```
TCanvas *ct = new TCanvas("Mass of Z boson", "mll",900,700);
```

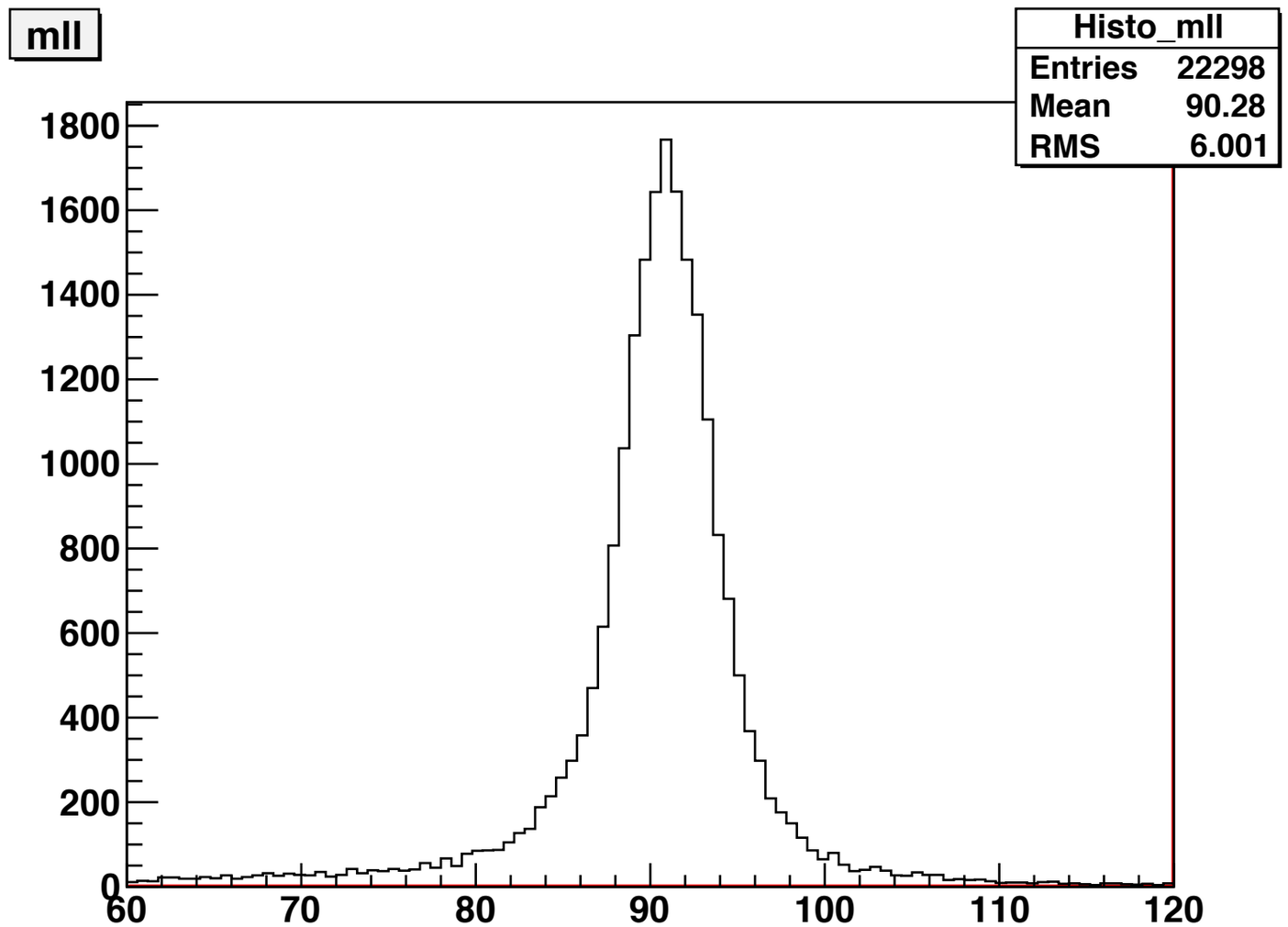
Name Title Δx Δy

  - Draw the histogram into the canvas

```
ct->cd(1);
Histo_mll->Draw();
```

Reconstructing M_Z

root -x Zmass.C+



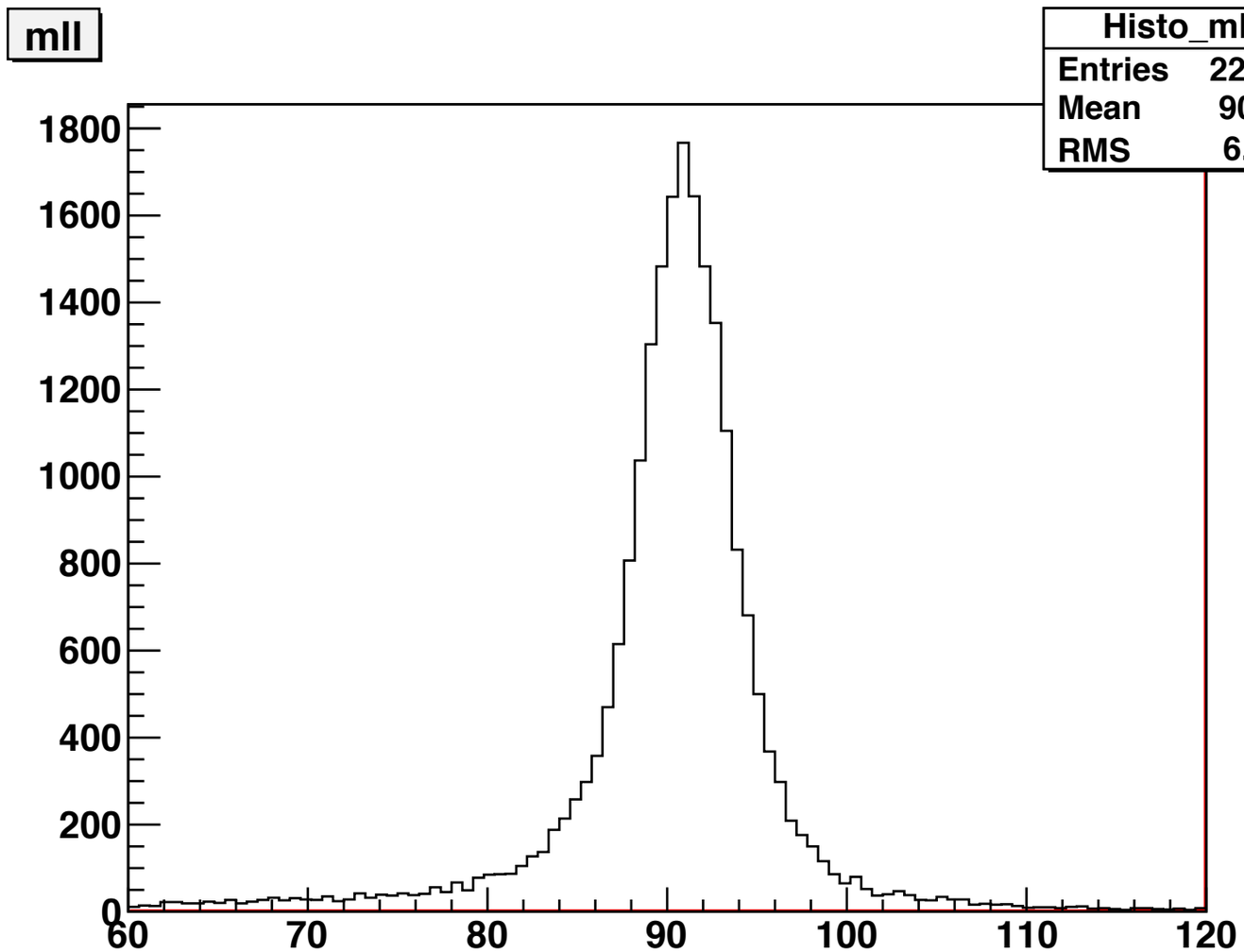
Where is the maximum ?

Reconstructing M_Z

- Little exercise : find the value of mll corresponding to the pic and write it on the plot

Reconstructing M_Z

- Little exercise : find the value of mll corresponding to the pic and write it on the plot



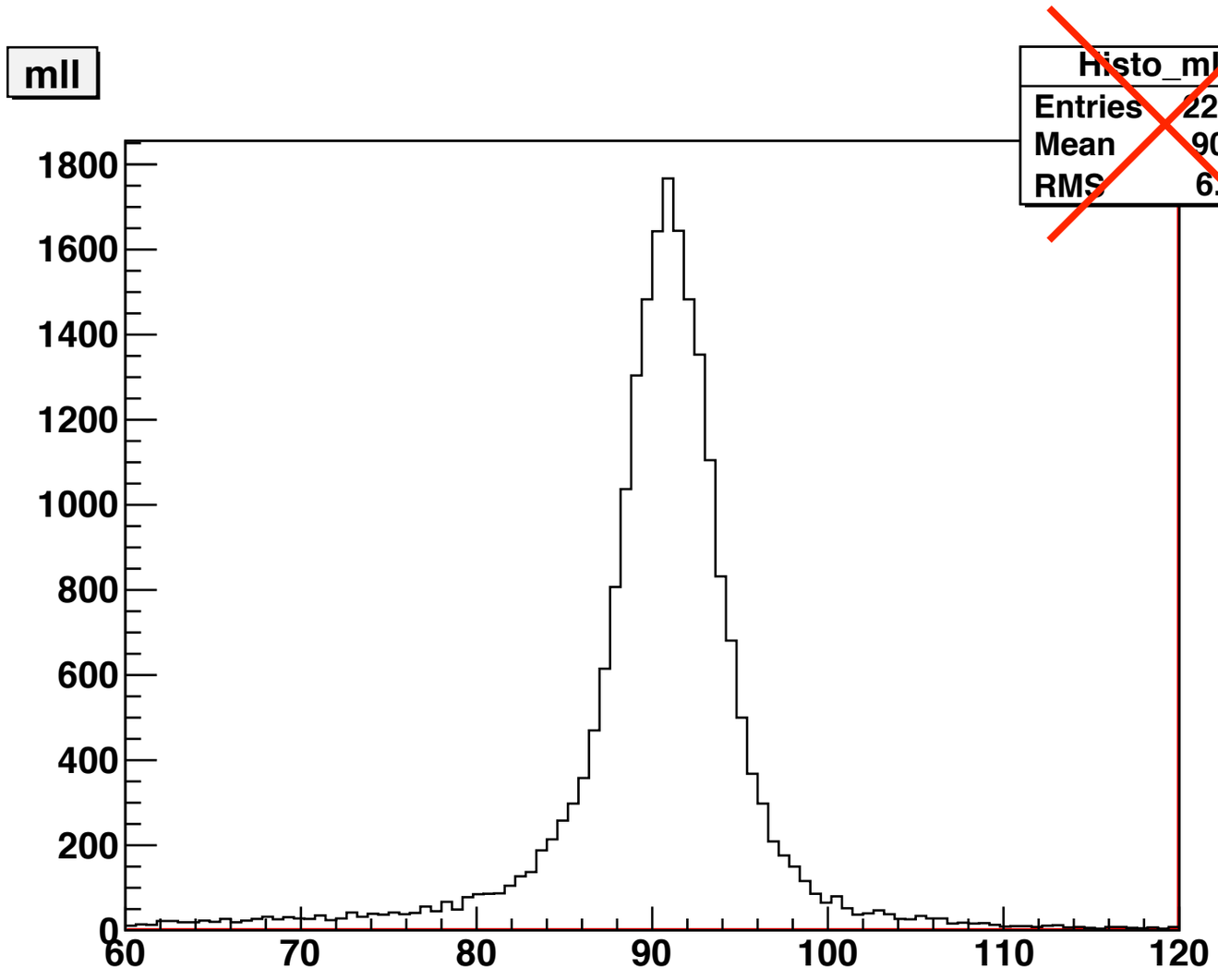
① Erase the statistic info.

② Find the maximum

③ Write a new legend

Reconstructing M_Z

- Little exercise : find the value of mll corresponding to the pic and write it on the plot

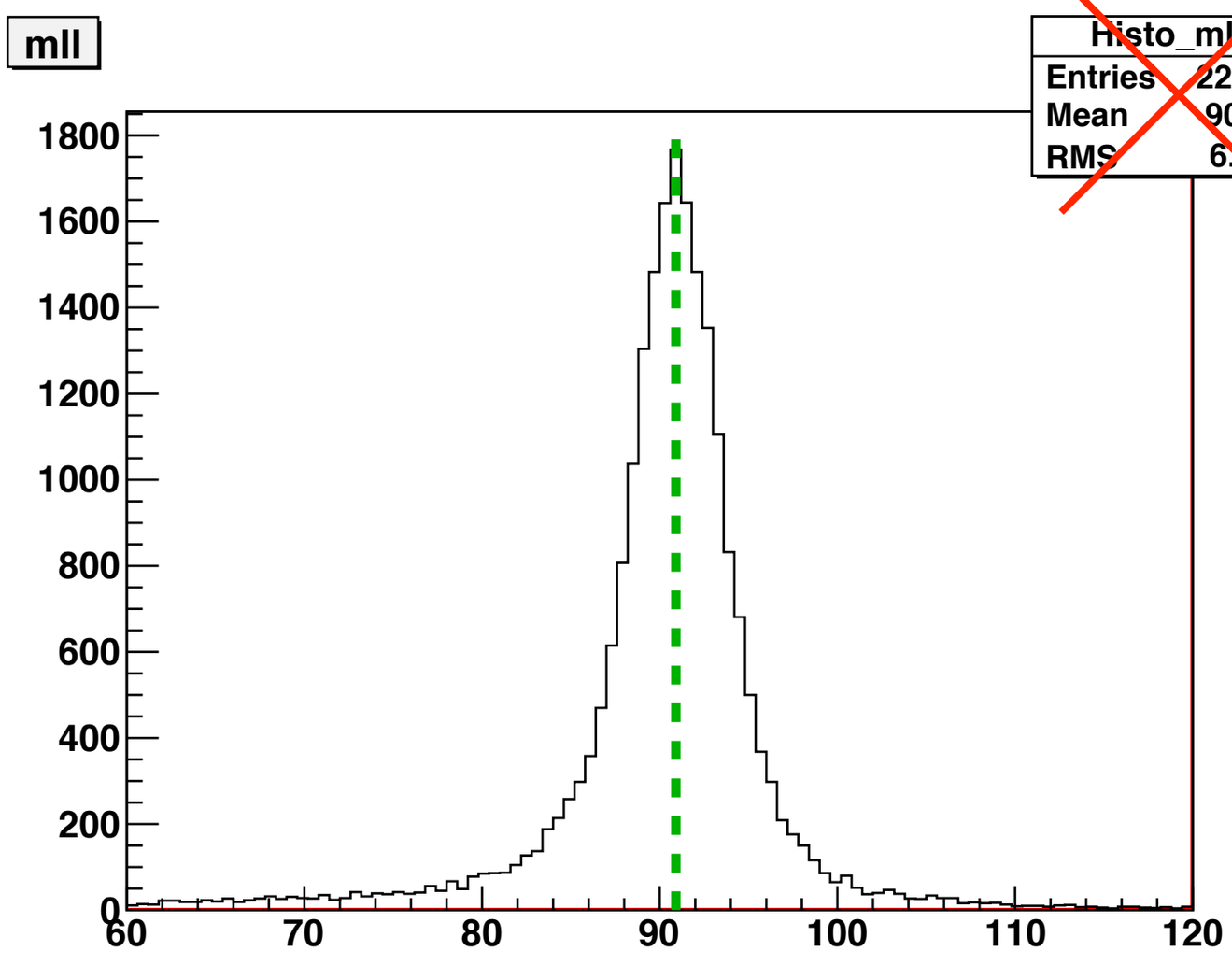


① Erase the statistic info.

```
Histo_mll->SetStats(0);
```

Reconstructing M_Z

- Little exercise : find the value of mll corresponding to the pic and write it on the plot



② Find the maximum

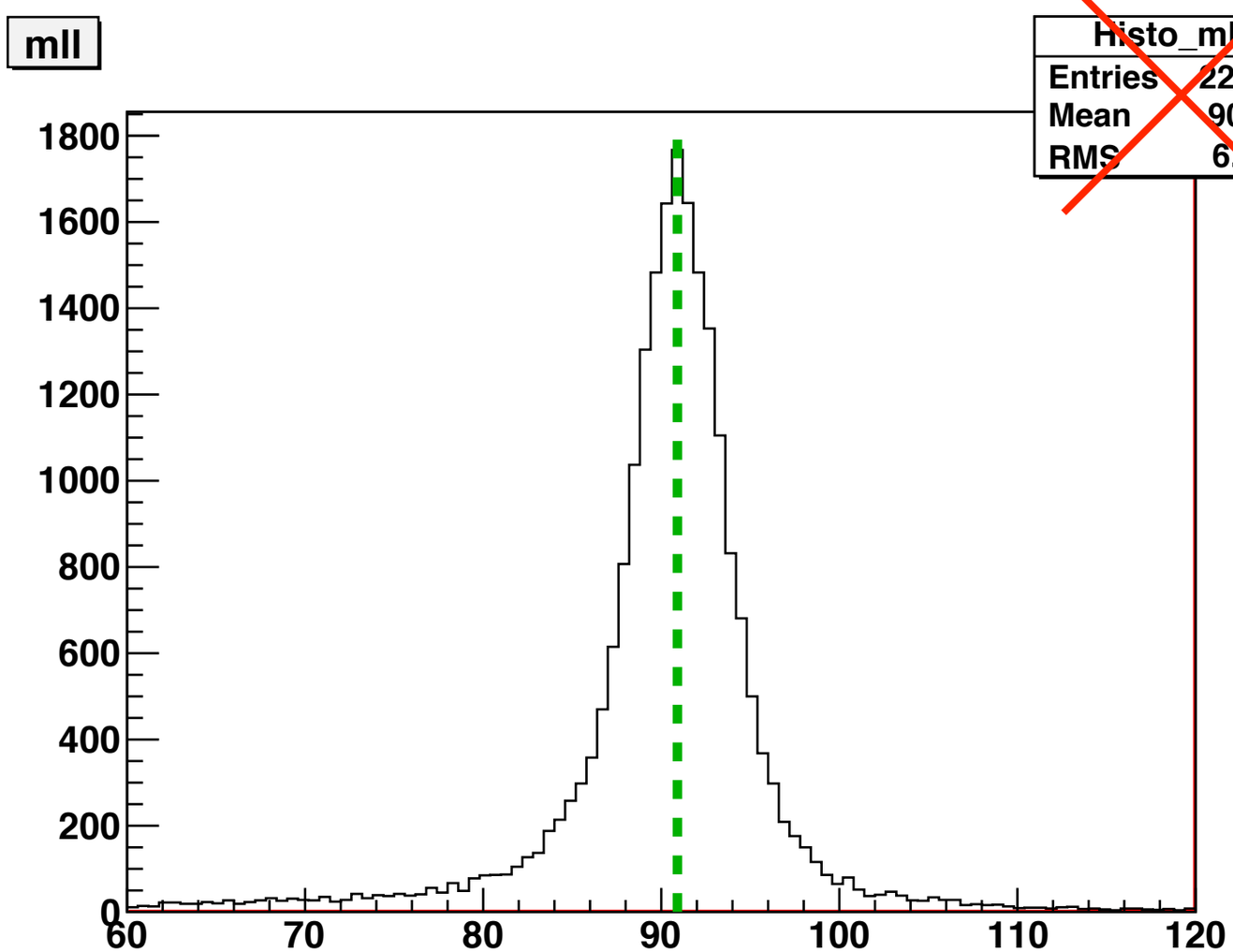
```
double max;
max = Histo_mll->GetMaximumBin();
```

100 bins between [60,120]

```
max*60/100+60
```

Reconstructing M_Z

- Little exercise : find the value of mll corresponding to the pic and write it on the plot



③ Write a new legend

Class TLegend existing

Reconstructing M_Z

- Little exercise : find the value of mll corresponding to the pic and write it on the plot

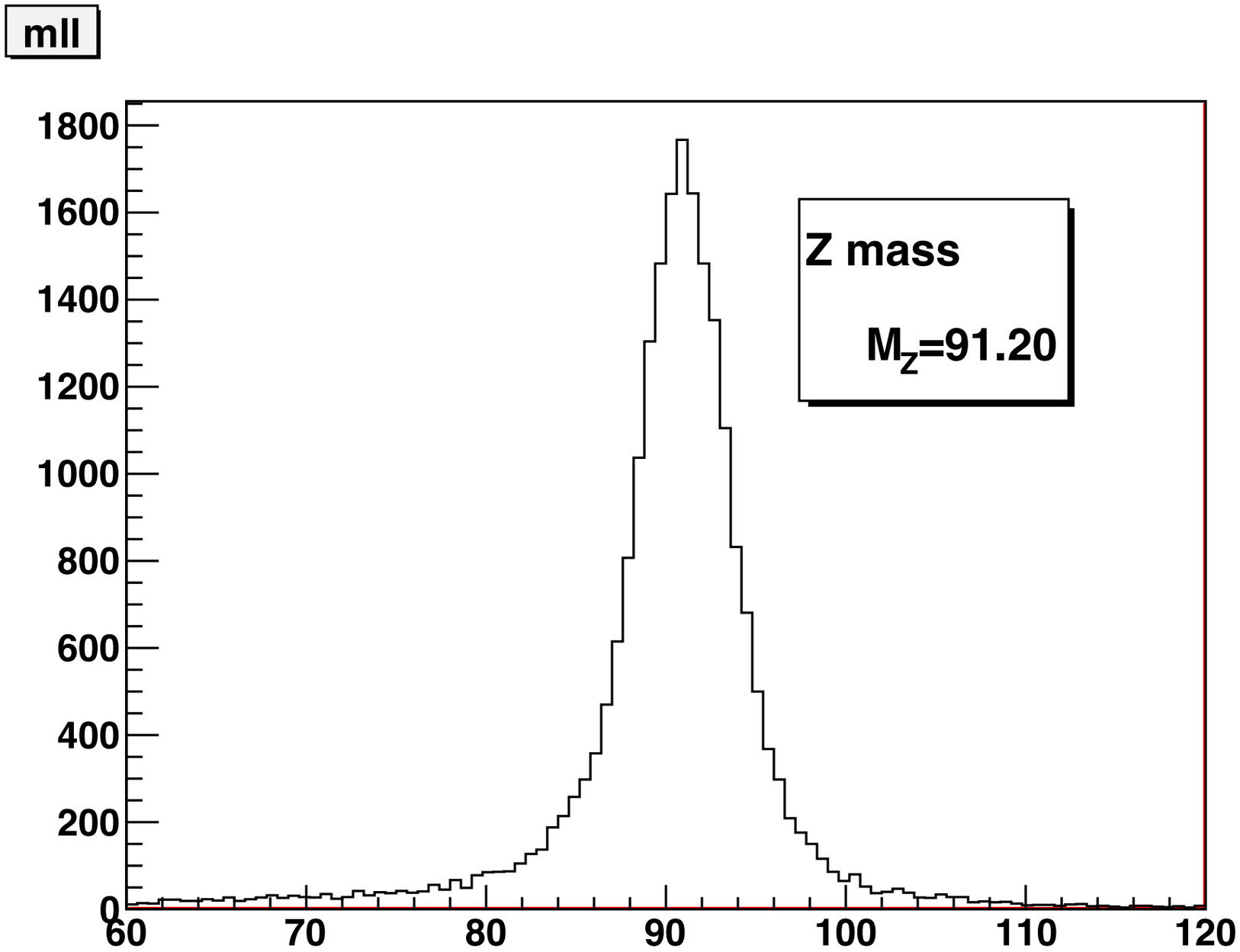
```
TLegend *leg = new TLegend(0.6,0.6,0.8,0.8,"Z mass");
```

Init. point Fin. point Header

```
Char_t aa[80];
sprintf(aa,"M_{Z}=%.2f",max*60/100+60);
leg->AddEntry((TObject*)0,aa,"");


ct->cd(1);
Histo_mll->SetStats(0);
Histo_mll->Draw();
leg->Draw();
```

Reconstructing M_Z

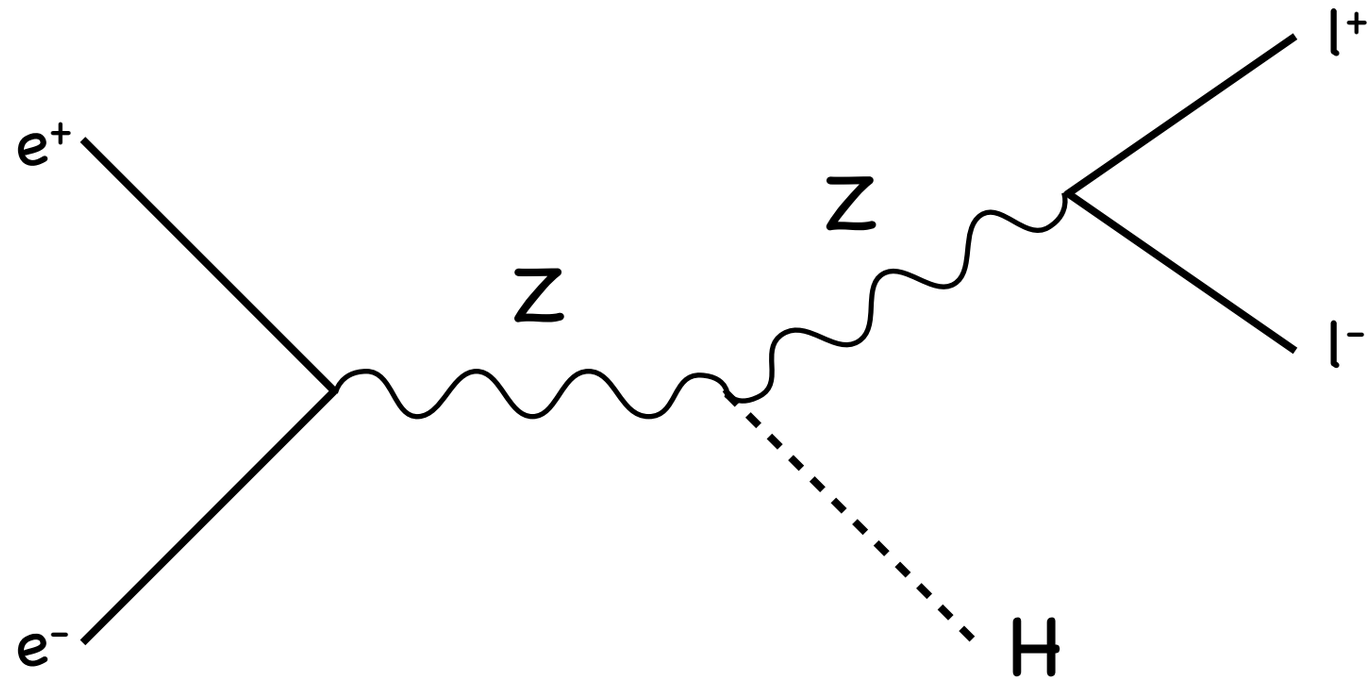
- Little exercise : find the value of mll corresponding to the pic and write it on the plot



2nd example :
Higgs at ILC

Higgs at ILC

- Use other functions of TLorentzVector
- Fit a distribution
- Continue with TLegend

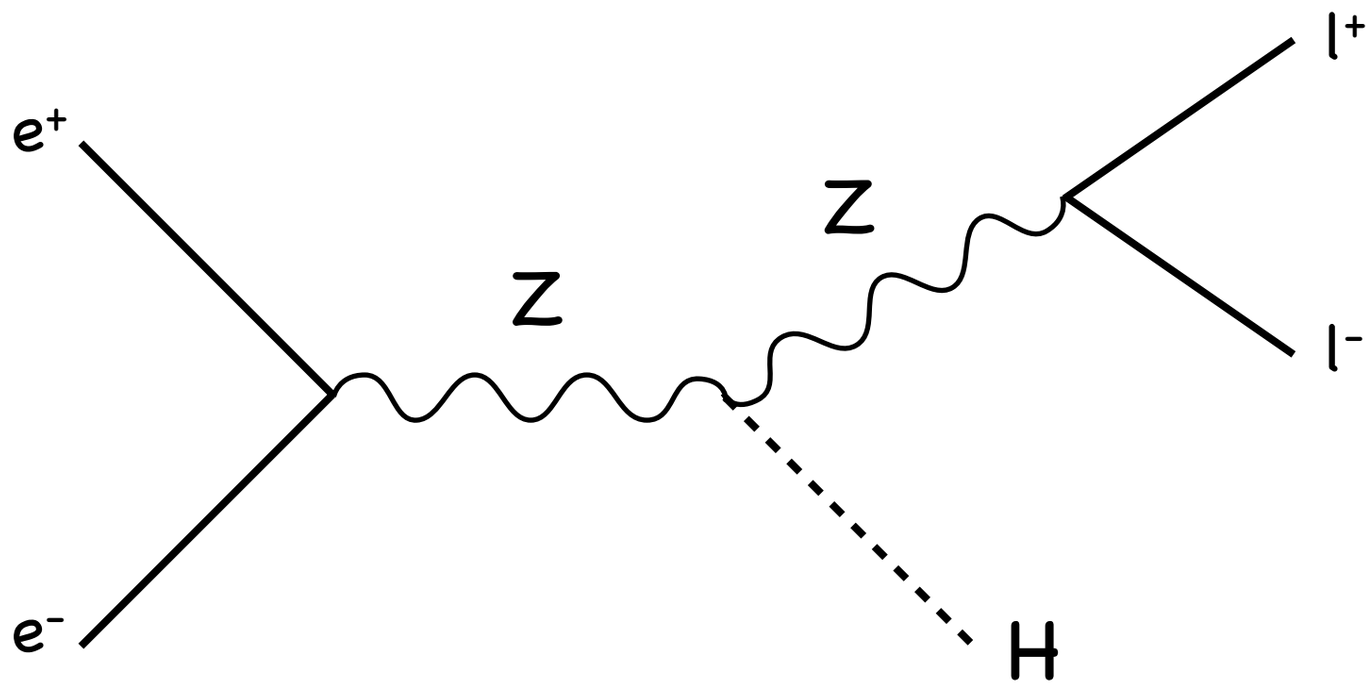


① Measure the Higgs mass without detecting its decays

② Look at angular distribution of leptons to see CP state of the scalar

Higgs at ILC

Recoil mass

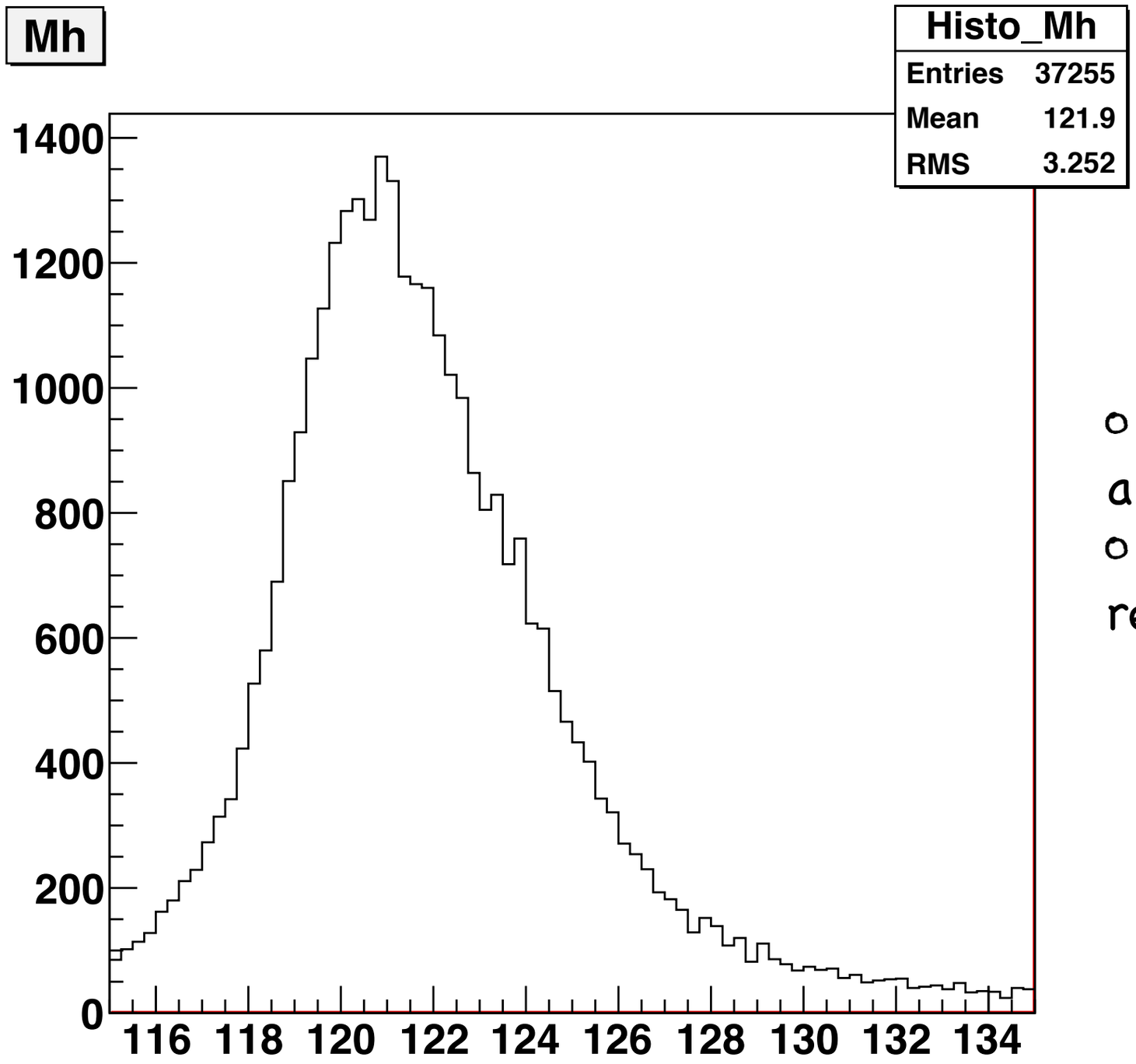


$$M_H^2 = s + M_Z^2 - 2E_Z\sqrt{s}$$

```
EZ = lepton1.E()+lepton2.E();
Mh = sqrt(250*250 + 91.2*91.2 - 2*EZ*250);
```

Higgs at ILC

Recoil mass



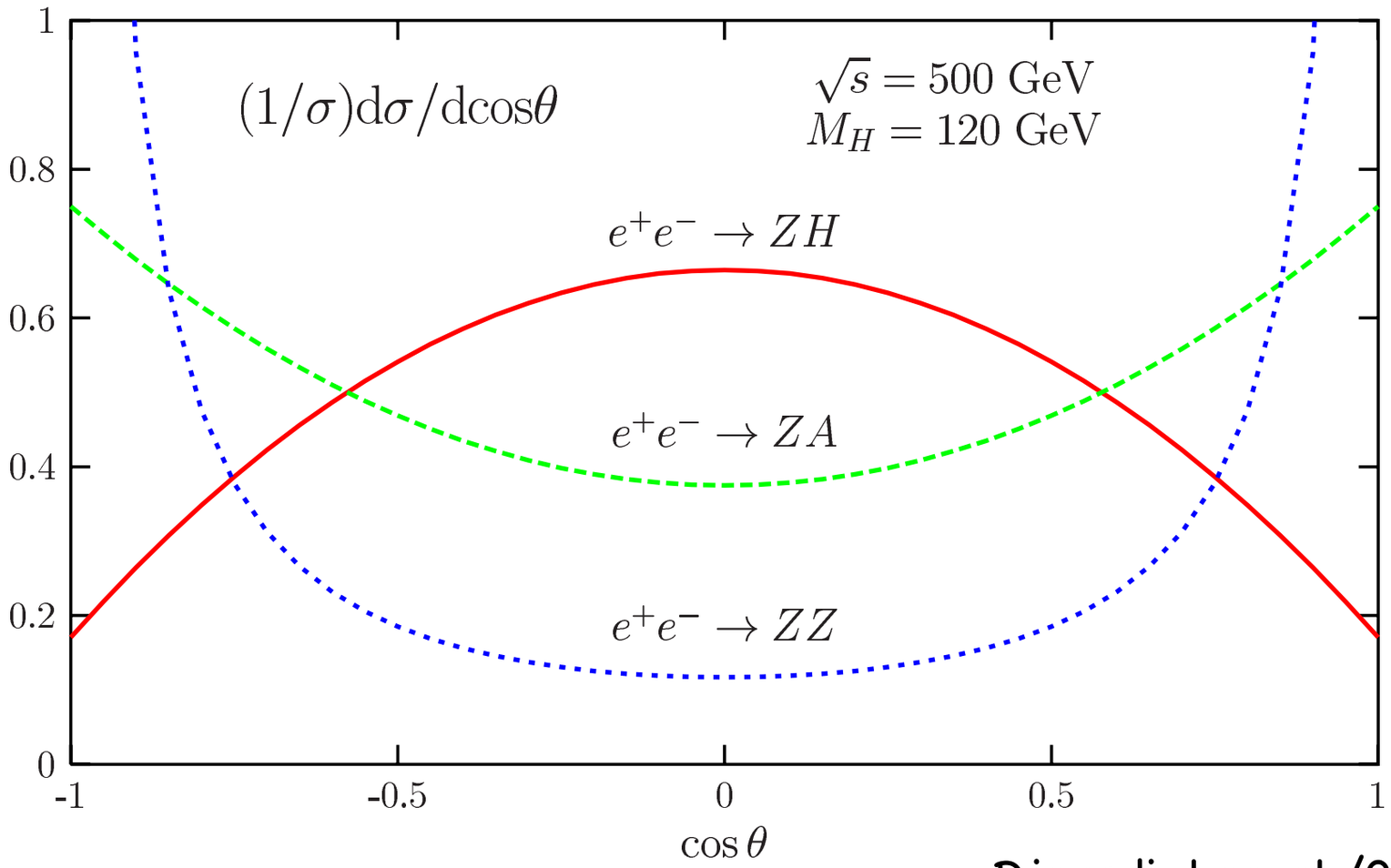
- Asymetric shape due to ISR and FSR
- Light shift due to detector response

Higgs at ILC

Spin/CP of the scalar

- Angular distribution of H is sensitive to the spin of H
- Spin 0, CP even $\propto$ $1-\cos^2\theta$
- Spin 0, CP odd $\propto$ $1+\cos^2\theta$

$s \gg M_Z^2$



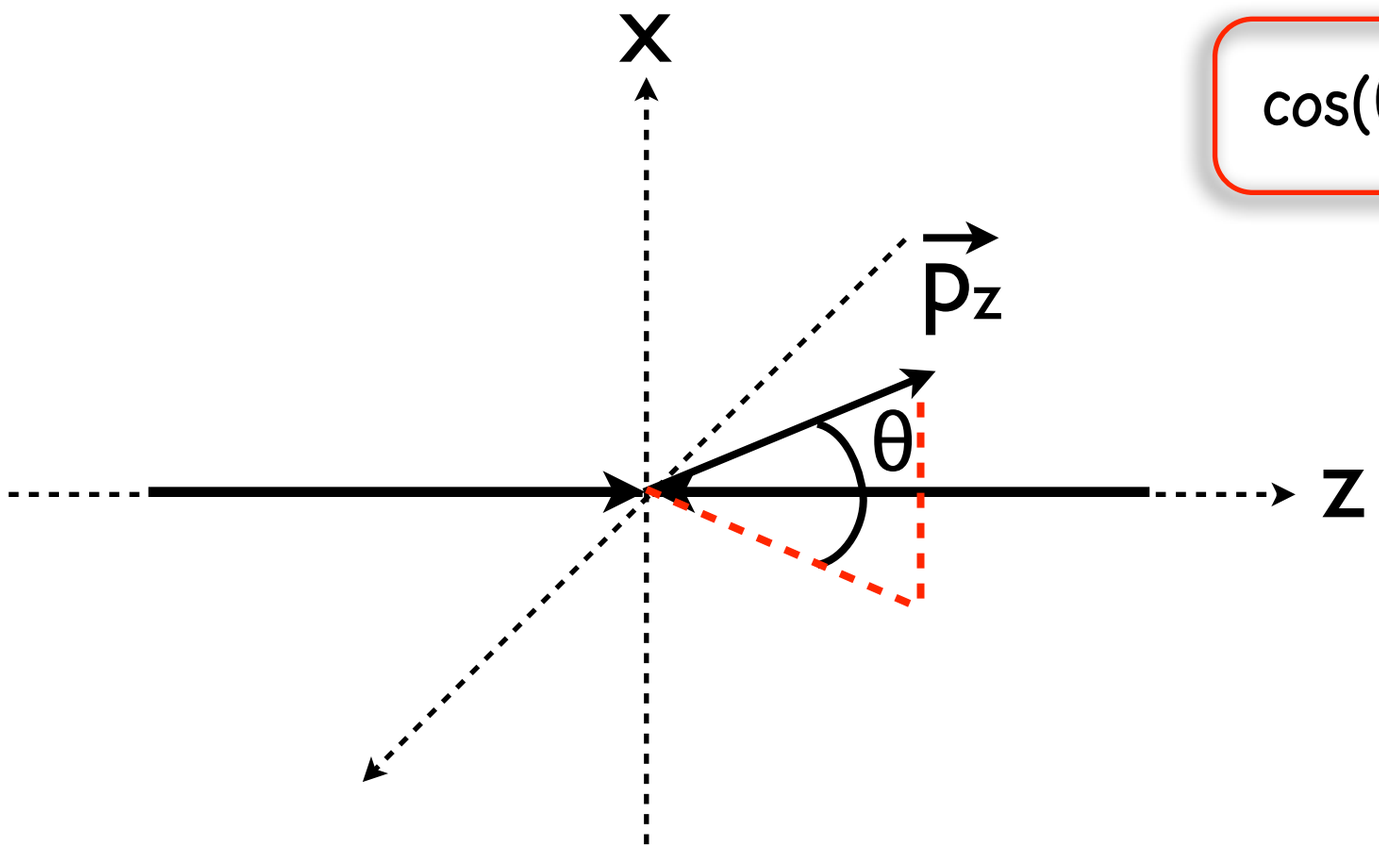
Djouadi, hep-ph/0503172

Higgs at ILC

Spin/CP of the scalar

- Angular distribution of Z/H is sensitive to the spin of H
- Spin 0, CP even $\propto 1-\cos^2\theta$
- Spin 0, CP odd $\propto 1+\cos^2\theta$

$s >> M_Z^2$



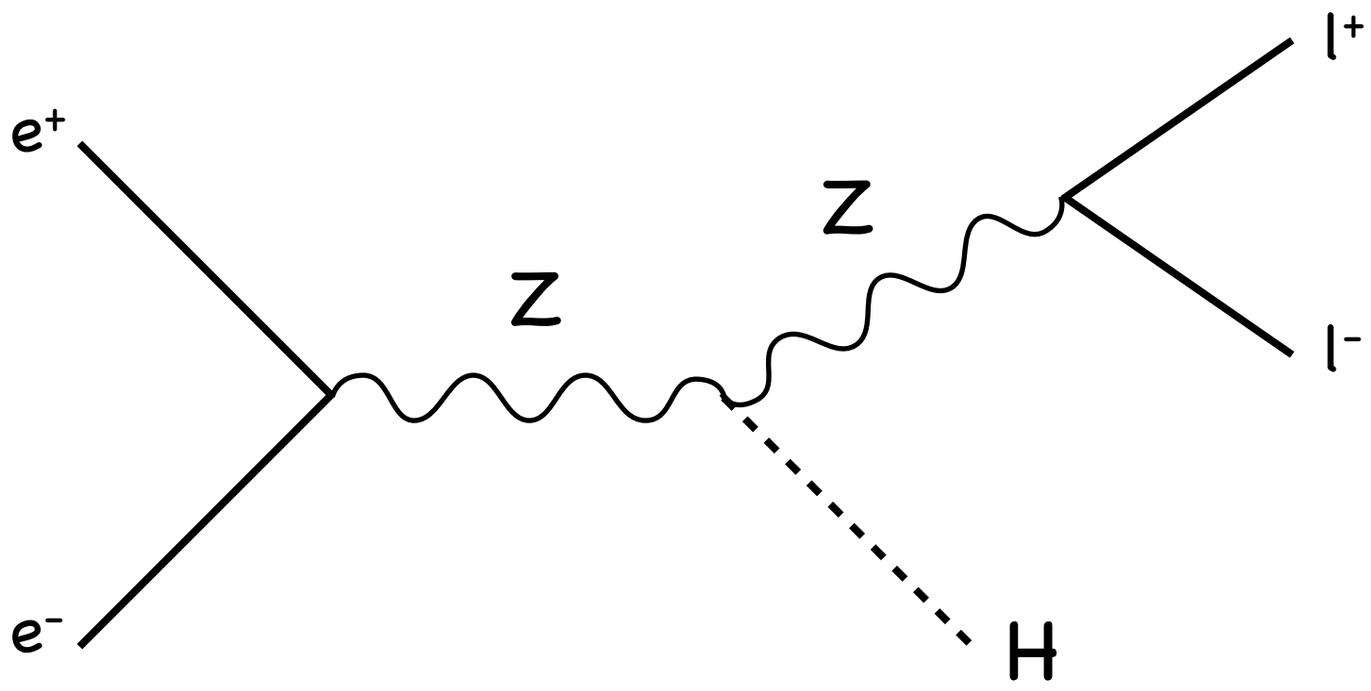
$\cos(\theta) = p_{zz} / |\vec{p}_z|$

Higgs at ILC

Spin/CP of the scalar

- Angular distribution of Z/H is sensitive to the spin of H
- Spin 0, CP even $\propto 1-\cos^2\theta$
- Spin 0, CP odd $\propto 1+\cos^2\theta$

$s >> M_Z^2$



$\cos(\theta) = p_{Zz} / |\vec{p_Z}|$

- $p_{Zz} = p_{l^+z} + p_{l^-z}$

- $\begin{cases} |\vec{p_Z}|^2 = E_Z^2 - M_Z^2 \\ |\vec{p_Z}|^2 = (p_{l^+x} + p_{l^-x})^2 + (p_{l^+y} + p_{l^-y})^2 + (p_{l^+z} + p_{l^-z})^2 \end{cases}$

Higgs at ILC

Spin/CP of the scalar

- Angular distribution of Z/H is sensitive to the spin of H
- Spin 0, CP even $\propto 1-\cos^2\theta$
- Spin 0, CP odd $\propto 1+\cos^2\theta$

$s \gg M_Z^2$

```
plz1 = lepton1.Pz();
plz2 = lepton2.Pz();
EZ = lepton1.E()+lepton2.E();
cth = (plz1+plz2)/sqrt(EZ*EZ-91.2*91.2);
```

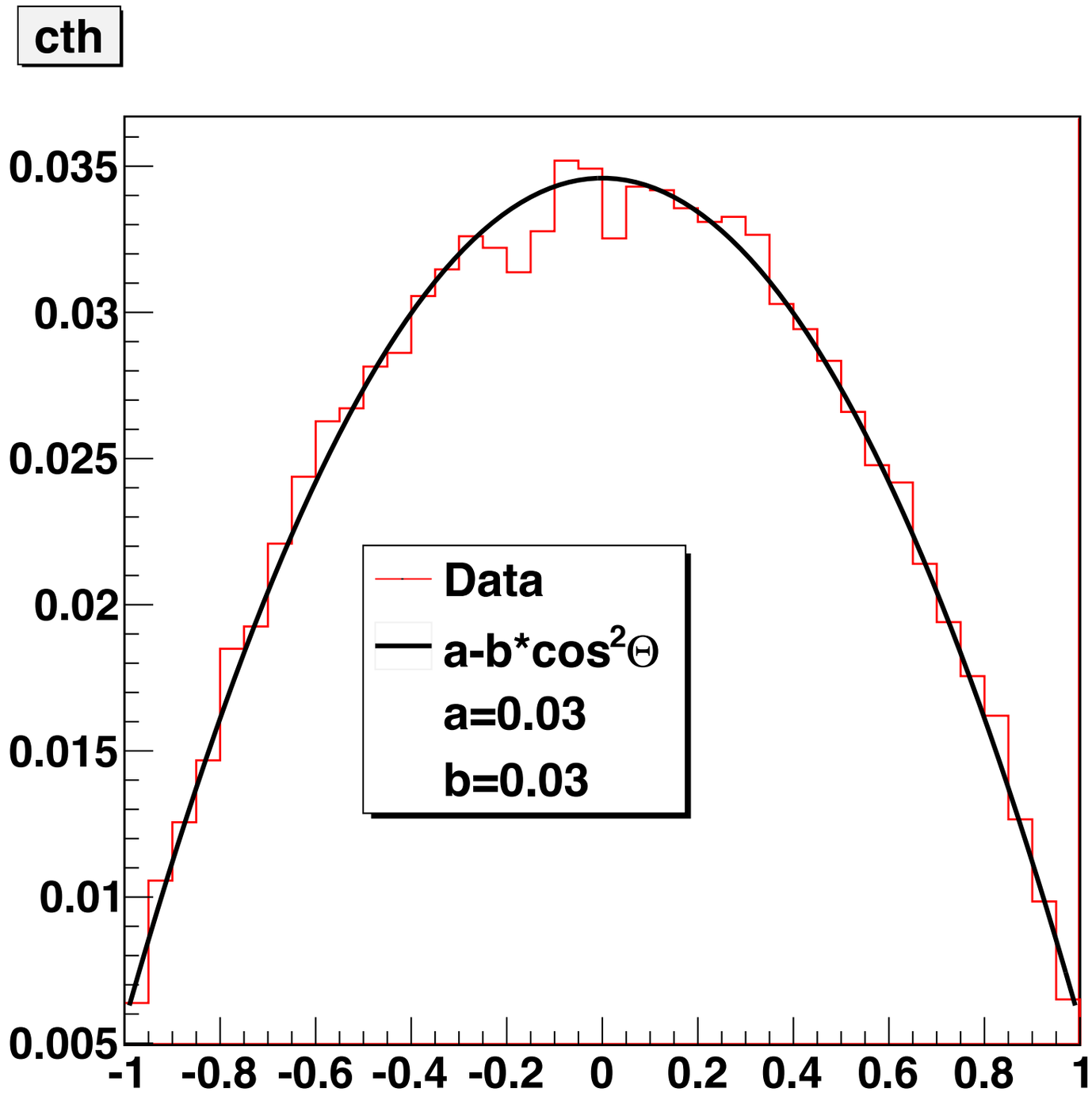
Declare, fill and draw the Histogram

$$\cos(\theta) = p_{Zz} \,/\, |\vec{p_Z}|$$

- $p_{Zz} = p_{l^+z} + p_{l^-z}$
- $$\begin{cases} |\vec{p_Z}|^2 = E_Z^2 - M_Z^2 \\ |\vec{p_Z}|^2 = (p_{l^+x} + p_{l^-x})^2 + (p_{l^+y} + p_{l^-y})^2 + (p_{l^+z} + p_{l^-z})^2 \end{cases}$$

Higgs at ILC

Spin/CP of the scalar



Higgs at ILC

- Normalizing the Histogram and changing the color

```
Double_t scale = 1/Histo_cth->Integral();
Histo_cth->Scale(scale);
Histo_cth->SetLineColor(kRed);
```

- Declaring a function

```
TF1 * f1 = new TF1("f1","[0]-[1]*x*x",-1,1);
```

parameters

- Fitting the data

```
f1->SetParameters(1,1);
Histo_cth->Fit("f1");
```

Higgs at ILC

- Preparing the Legend

no header

```
TLegend *leg = new TLegend(0.3,0.3,0.57,0.53);

TF1 *fun = Histo_cth->GetFunction("f1");

sprintf(aa,"a=%.2f",fun->GetParameter(0));
sprintf(bb,"b=%.2f",fun->GetParameter(1));

leg->AddEntry(Histo_cth,"Data");
leg->AddEntry(f1,"a-b*cos^{2}#Theta");
leg->AddEntry((TObject*)0,aa,"");
leg->AddEntry((TObject*)0,bb,"");
```

Higgs at ILC

- Preparing the Legend

```
TLegend *leg = new TLegend(0.3,0.3,0.57,0.53);

TF1 *fun = Histo_cth->GetFunction("f1");

sprintf(aa,"a=%.2f",fun->GetParameter(0));
sprintf(bb,"b=%.2f",fun->GetParameter(1));

leg->AddEntry(Histo_cth,"Data");
leg->AddEntry(f1,"a-b*cos^{2}#Theta");
leg->AddEntry((TObject*)0,aa,"");
leg->AddEntry((TObject*)0,bb,"");
```

The fit had been store in the Histogram

Higgs at ILC

- Preparing the Legend

```
TLegend *leg = new TLegend(0.3,0.3,0.57,0.53);

TF1 *fun = Histo_cth->GetFunction("f1");

sprintf(aa,"a=%.2f",fun->GetParameter(0));
sprintf(bb,"b=%.2f",fun->GetParameter(1));

leg->AddEntry(Histo_cth,"Data");
leg->AddEntry(f1,"a-b*cos^{2}#Theta");
leg->AddEntry((TObject*)0,aa,"");
leg->AddEntry((TObject*)0,bb,"");
```

→ Read the parameters

Higgs at ILC

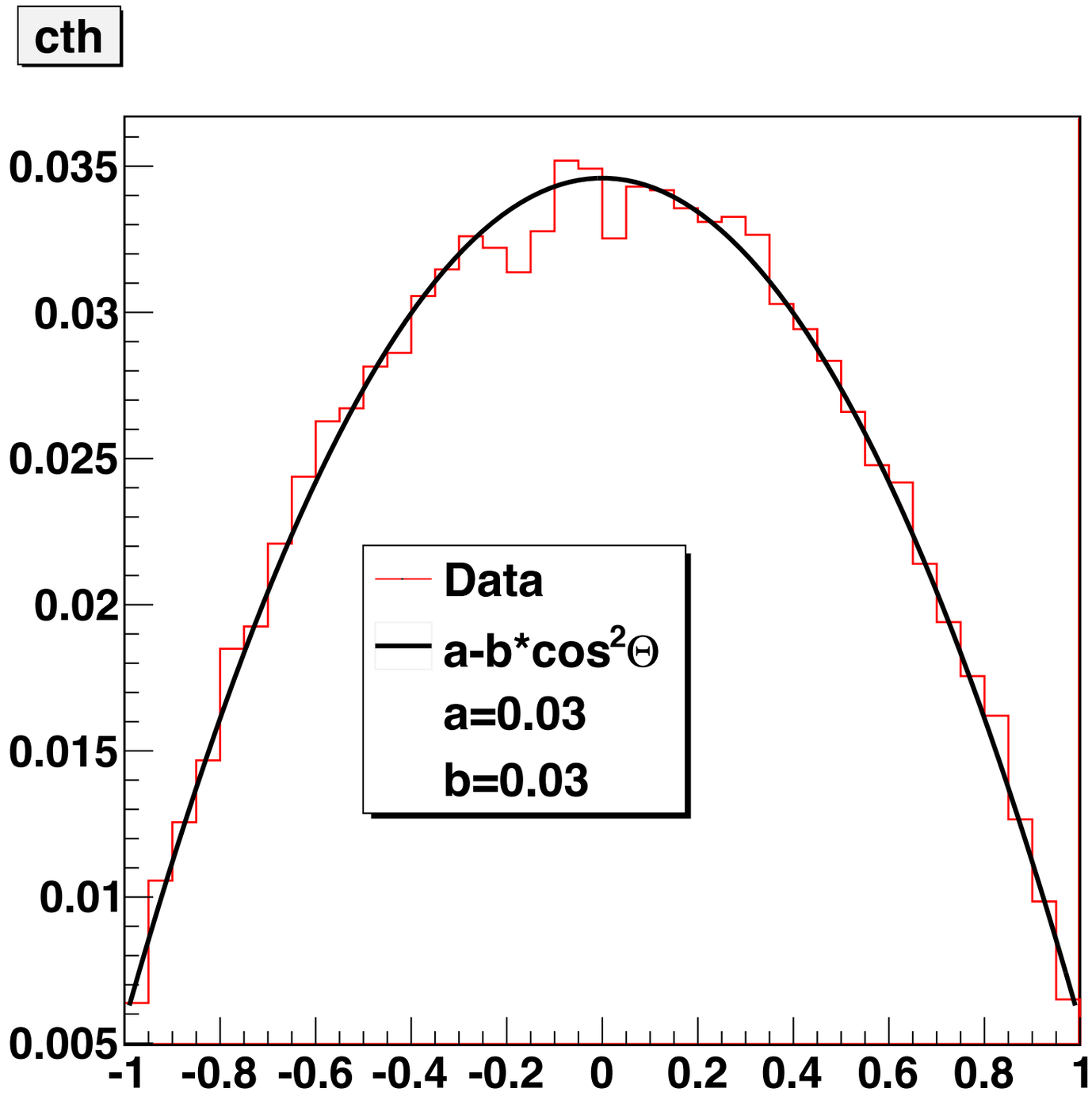
- Preparing the Legend

```
TLegend *leg = new TLegend(0.3,0.3,0.57,0.53);

TF1 *fun = Histo_cth->GetFunction("f1");

sprintf(aa,"a=%.2f",fun->GetParameter(0));
sprintf(bb,"b=%.2f",fun->GetParameter(1));

leg->AddEntry(Histo_cth,"Data");
leg->AddEntry(f1,"a-b*cos^{2}#Theta");
leg->AddEntry((TObject*)0,aa,"");
leg->AddEntry((TObject*)0,bb,"");
```

Adjust the type of line to the style of the Histogram or the function

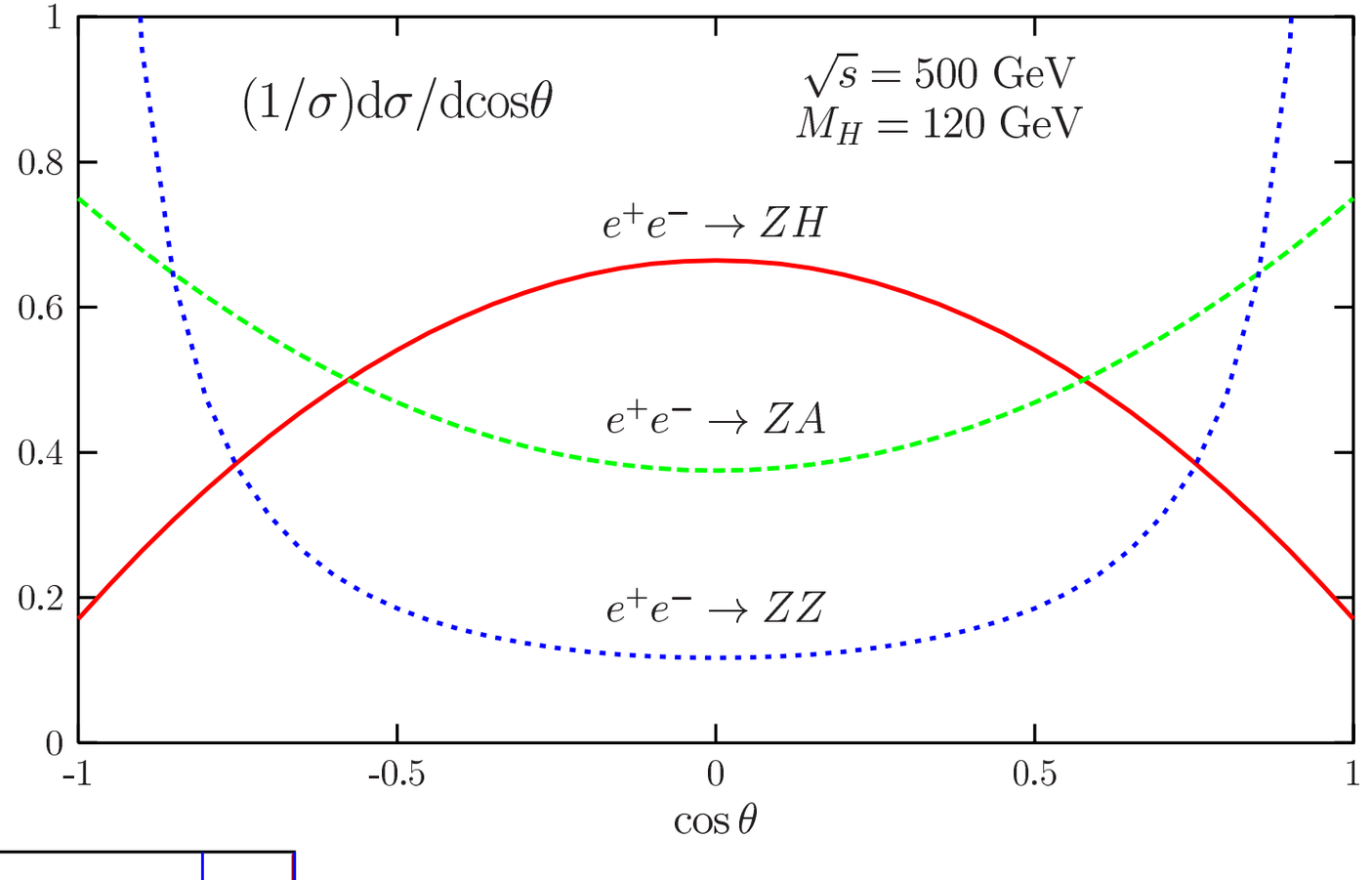
Higgs at ILC

Spin/CP of the scalar

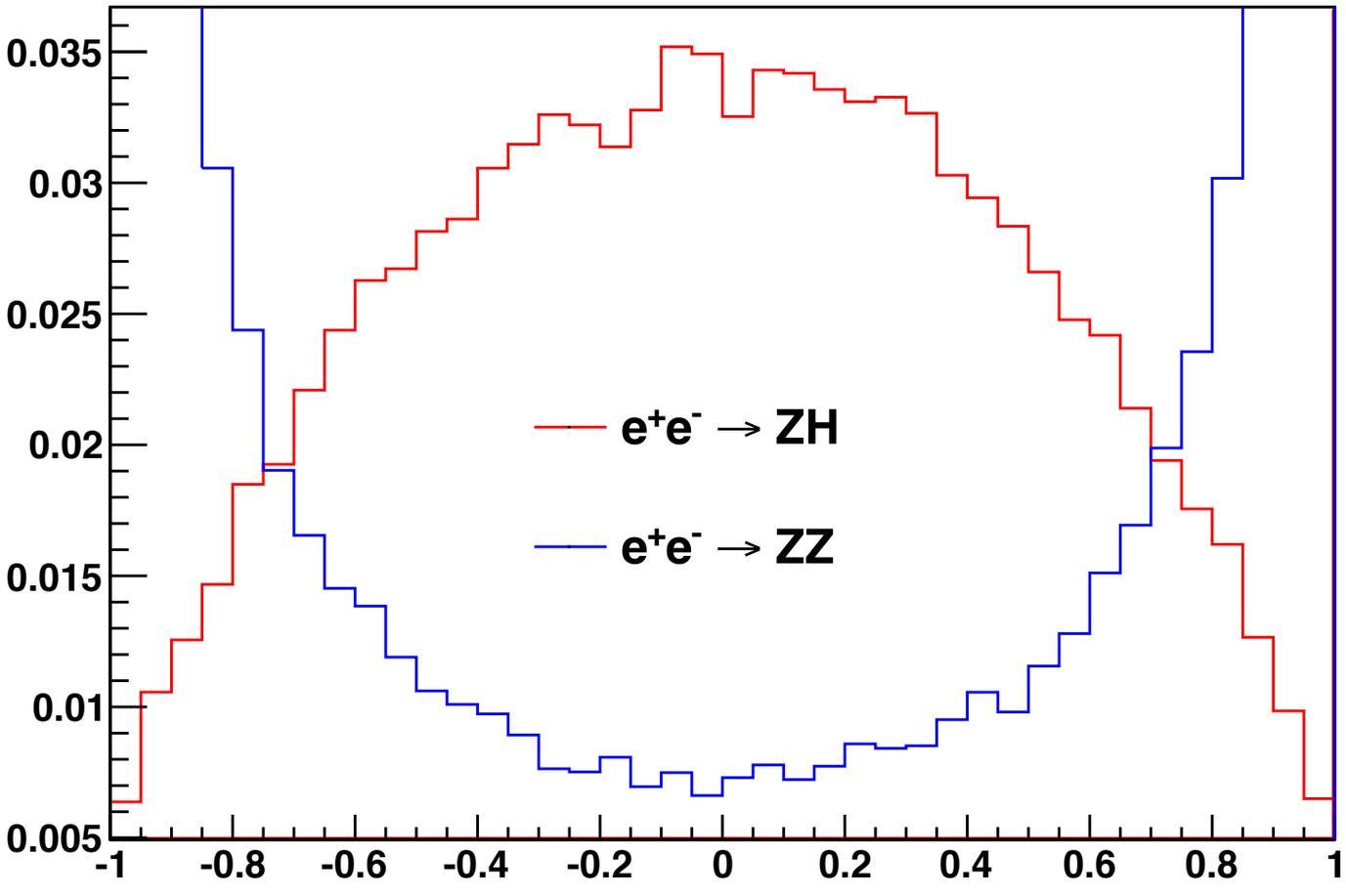


Higgs at ILC

Spin/CP of the scalar



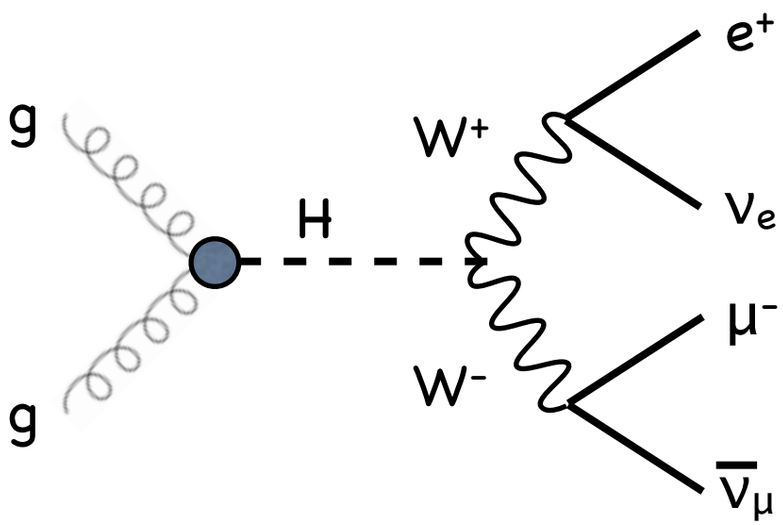
cth

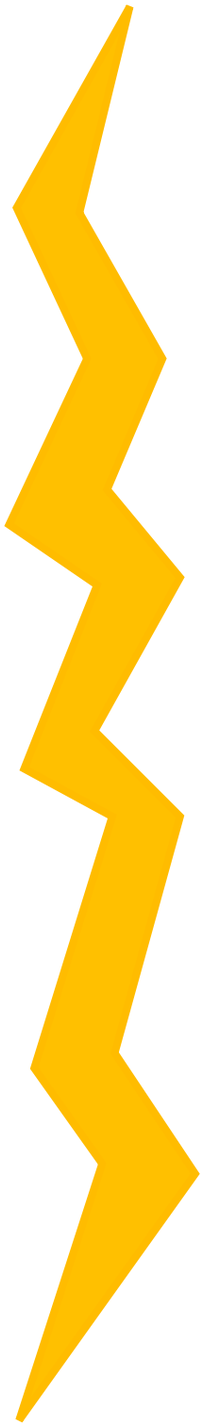


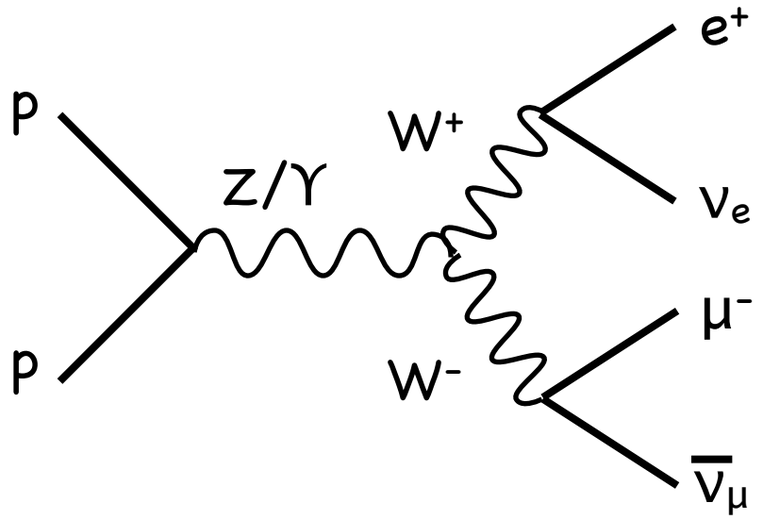
3rd example :

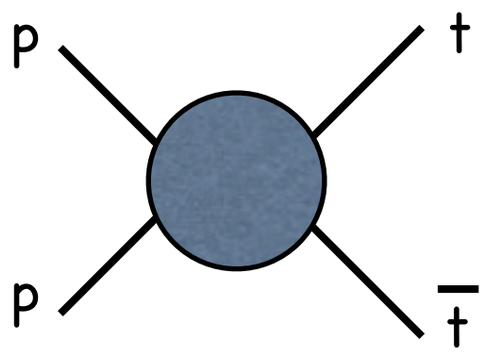
Signal Vs Background

Signal Vs Background









Signal Vs Background

- Study 3 files at a time

```
const int nSims = 3;
char * fileNames[nSims] = {"ggHww_pgs_events.lhco","reduc_pgs_events.lhco","irred_pgs_events.lhco"};
```

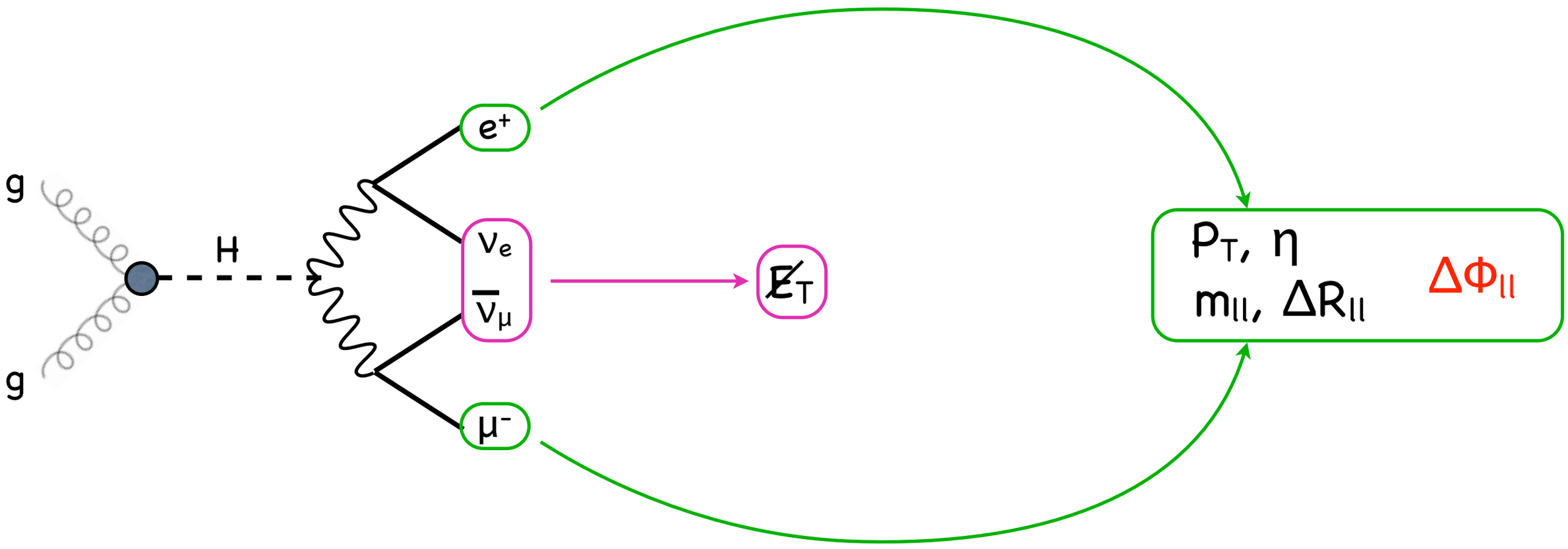
⇒ loop over the files

```
for (int simIt=0; simIt<3; simIt++){
```

- Use all relevant kinematic observables to discriminate signal and backgrounds :

Which ones ?

Signal Vs Background



Signal Vs Background

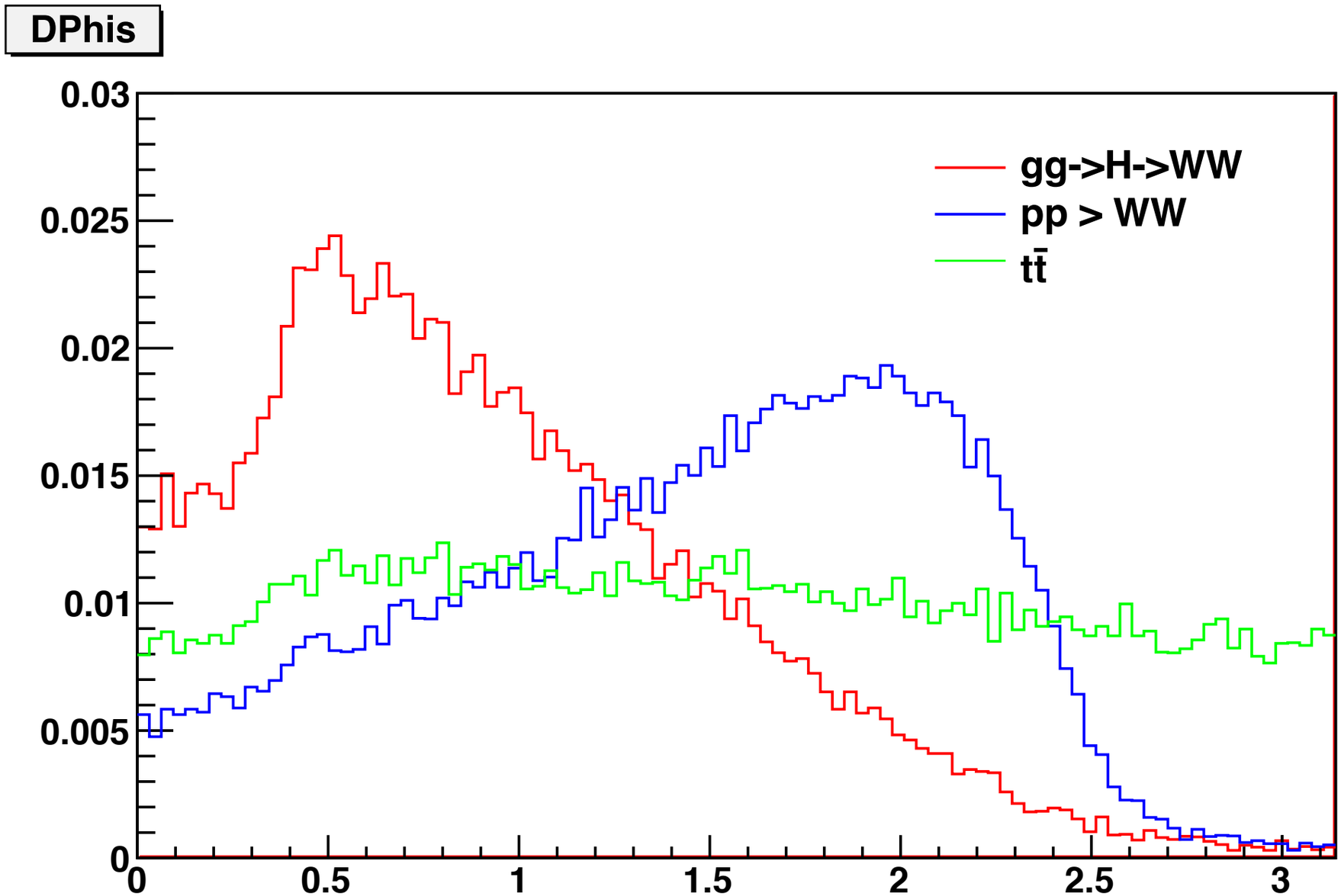
Sort leptons by P_T

```
MissET = MET[0].pt;

sort(leptons.begin(), leptons.end(), comparePt);
lepton1.SetPtEtaPhiM(leptons[0].pt,leptons[0].eta,leptons
[0].phi,leptons[0].jma);
lepton2.SetPtEtaPhiM(leptons[1].pt,leptons[1].eta,leptons
[1].phi,leptons[1].jma);

ptl1 = lepton1.Pt();
ptl2 = lepton2.Pt();
etal1 = lepton1.Eta();
etal2 = lepton2.Eta();
DRll = lepton1.DeltaR(lepton2);
mll = (lepton1 + lepton2).M();


DPhi = fabs(lepton1.DeltaPhi(lepton2));
```

Signal Vs Background

```
if (simIt==0) {
        Histo_Dphis->Fill(DPhi);
        Histo_MET->Fill(MissET);
        Histo_ptl1->Fill(ptl1);
        Histo_ptl2->Fill(ptl2);
        Histo_etal1->Fill(etal1);
        Histo_etal2->Fill(etal2);
        Histo_DRll->Fill(DRll);
        Histo_mll->Fill(mll);
}
```

Signal Vs Background



Signal Vs Background

- Normalize the histogram

```
Double_t scaleXXX = 1/Histo_XXX->Integral();
Histo_XXX->Scale(scaleXXX);
```

- Legend with corresponding lines

```
TLegend *leg = new TLegend(0.6,0.6,0.87,0.83);
leg->AddEntry(Histo_Dphis,"gg->H->WW","l");
leg->AddEntry(Histo_Dphiirred,"pp > WW","l");
leg->AddEntry(Histo_Dphireduc,"t#bar{t}","l");
```

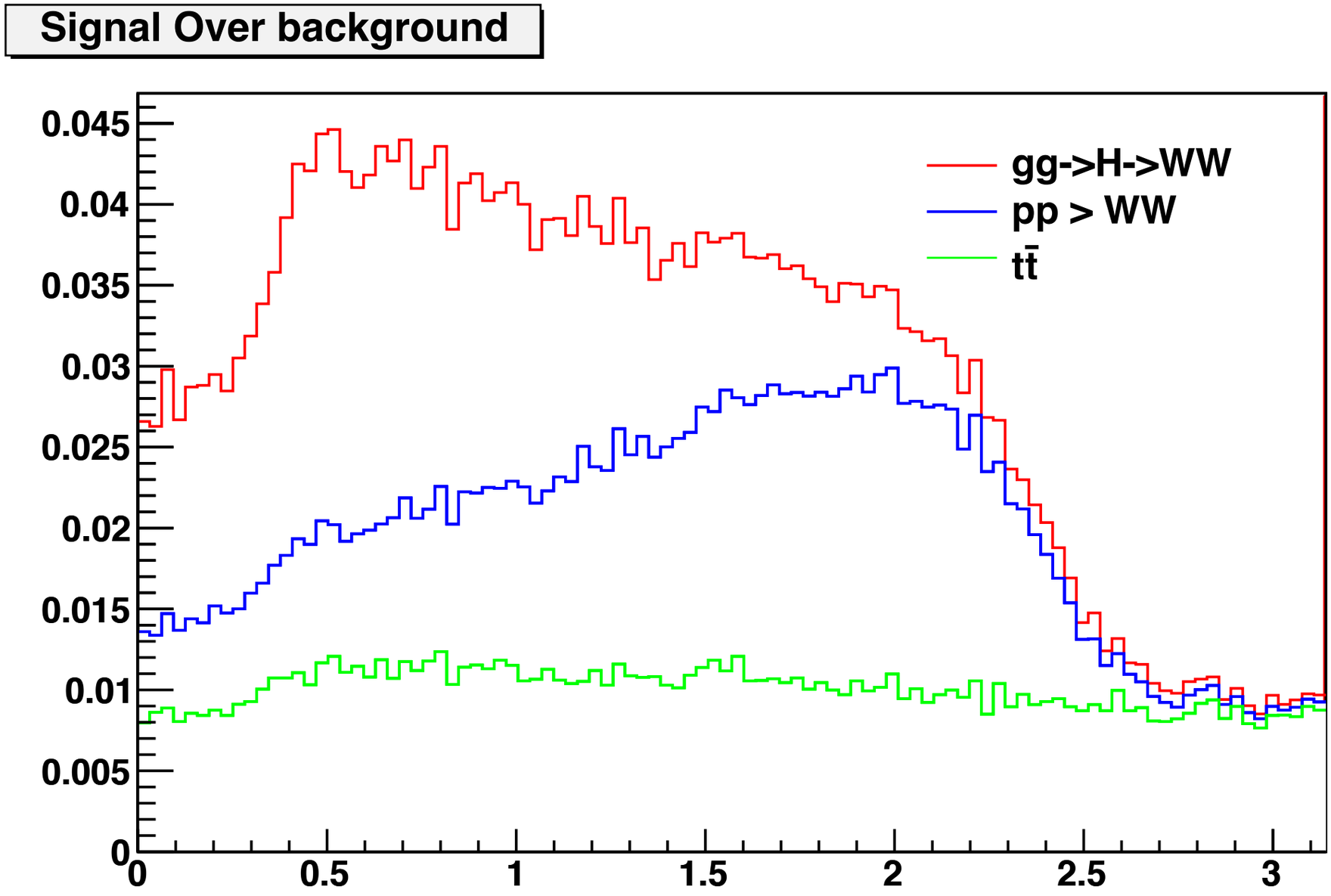
- Set the color

```
Histo_Dphis->SetLineColor(kRed);
```

- Plot everything in the same canvas slot

```
ct3->cd(1);
Histo_Dphis->Draw();
Histo_Dphiirred->Draw("same");
Histo_Dphireduc->Draw("same");
leg->Draw("same");
```

Signal Vs Background

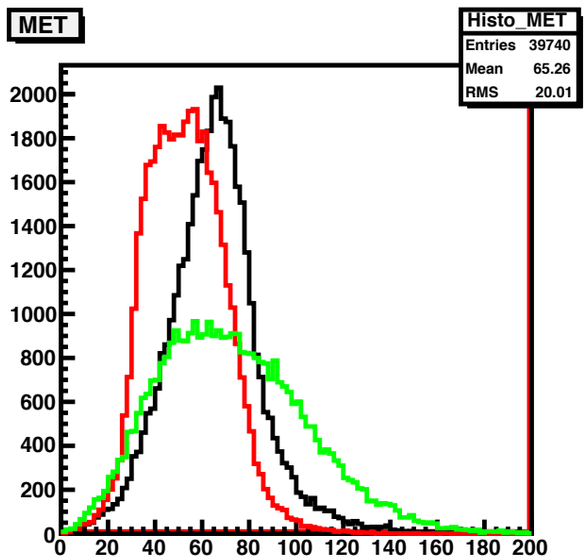


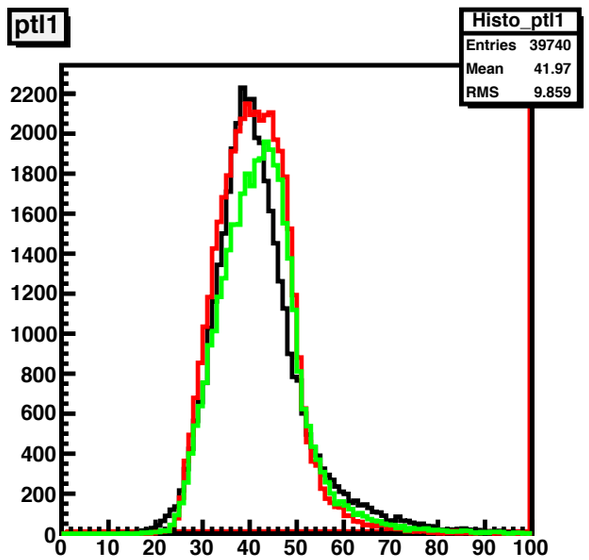
Signal Vs Background

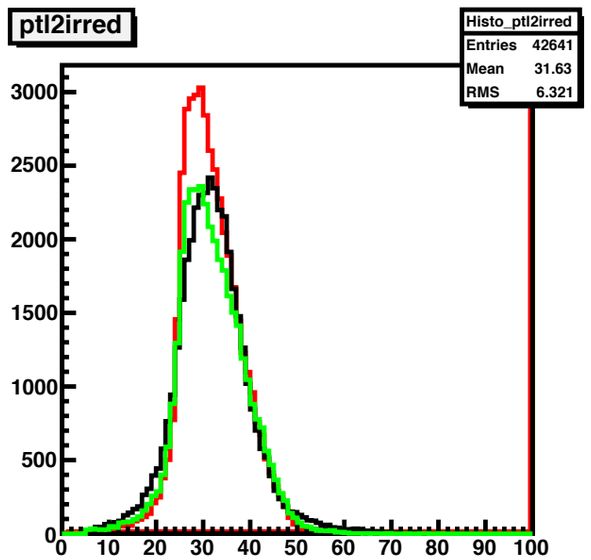
```
THStack Histo_stack("Histo_stack","Signal Over background");
```

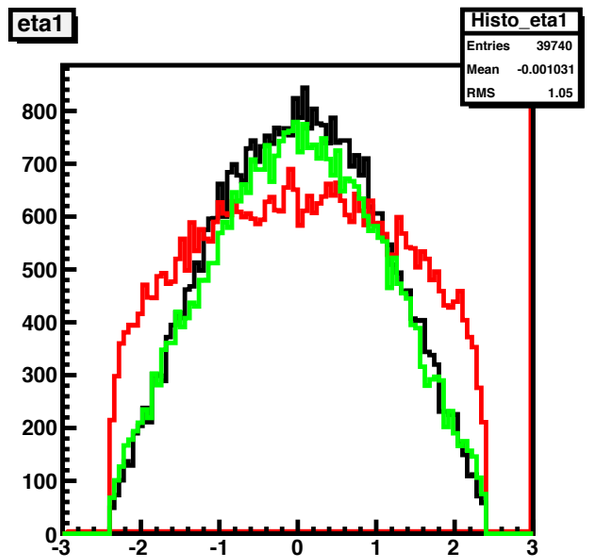
```
ct2->cd(1);
Histo_stack.Add(Histo_Dphireduc);
Histo_stack.Add(Histo_Dphiirred);
Histo_stack.Add(Histo_Dphis);
Histo_stack.Draw();
leg->Draw("same");
```

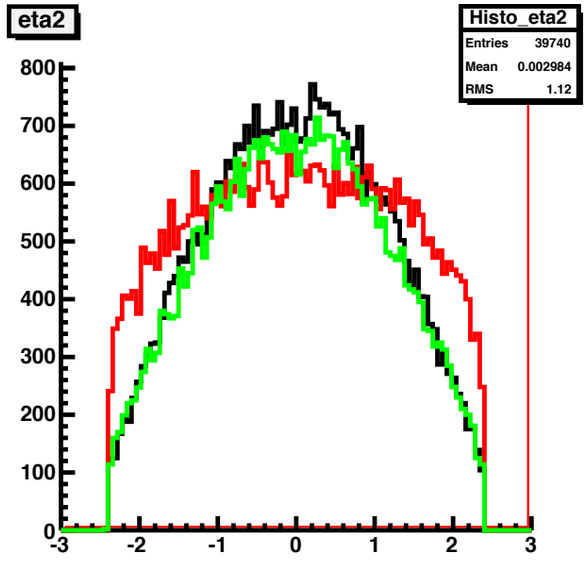
Signal Vs Background

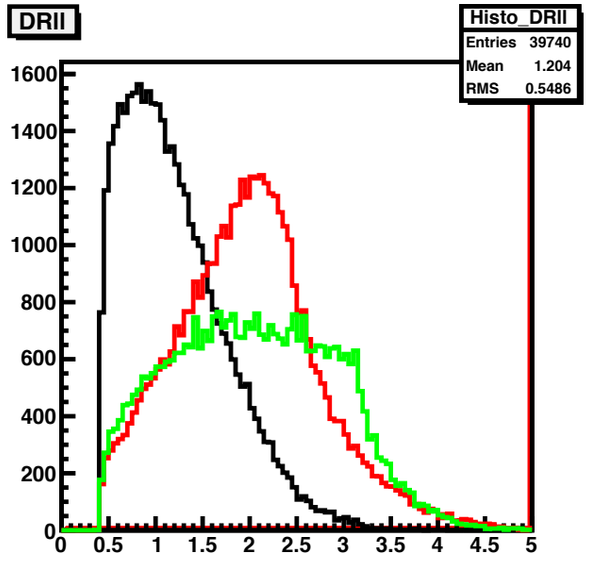


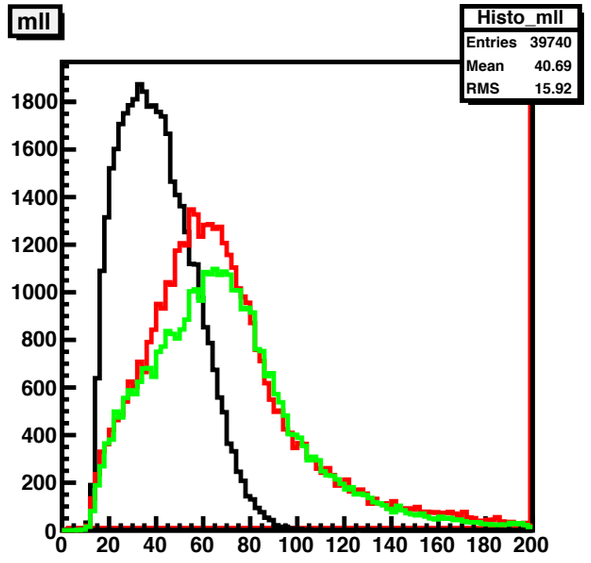












Signal Vs Background

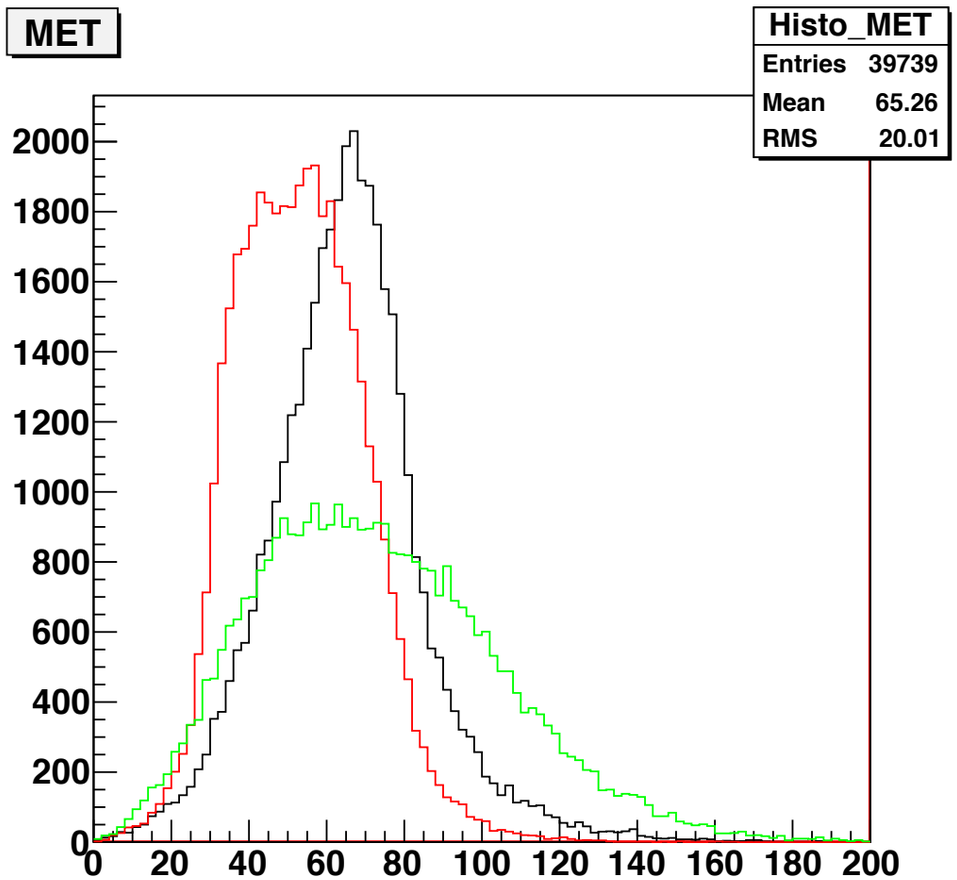
Acceptance cuts

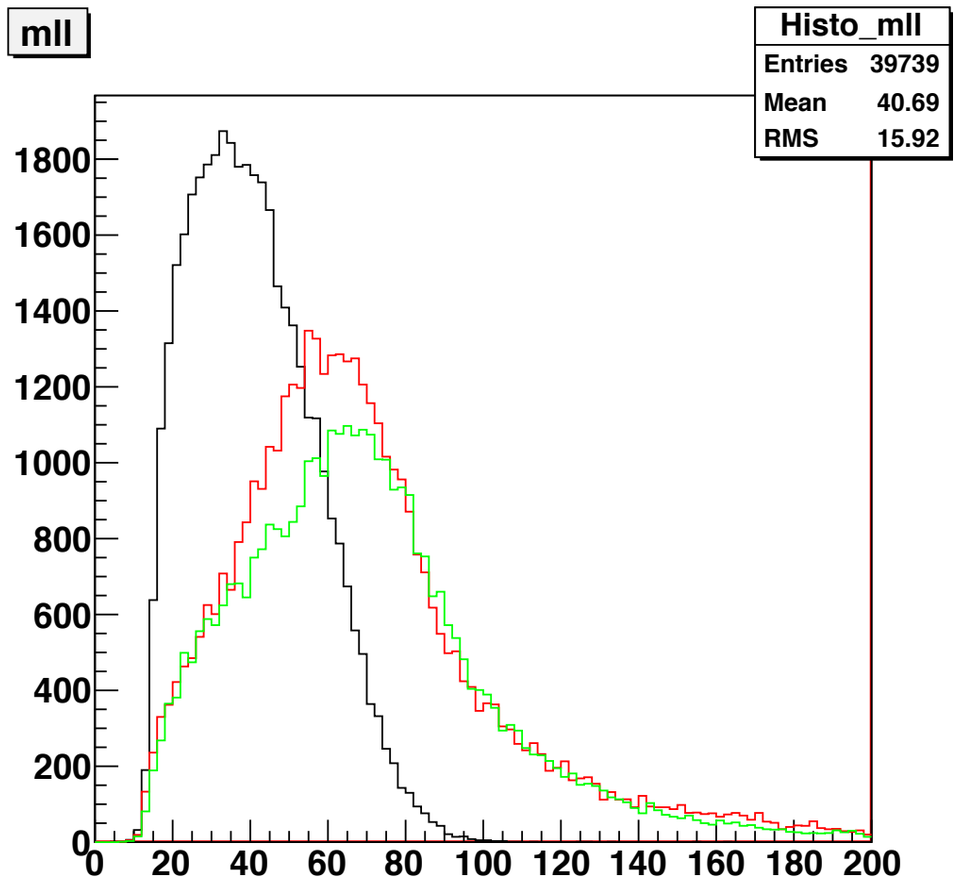
```
if (fabs(etal1) > 2.4) {
  continue;
}
if (fabs(etal2) > 2.4) {
  continue;
}
if (DRll < 0.4) {
  continue;
}
```

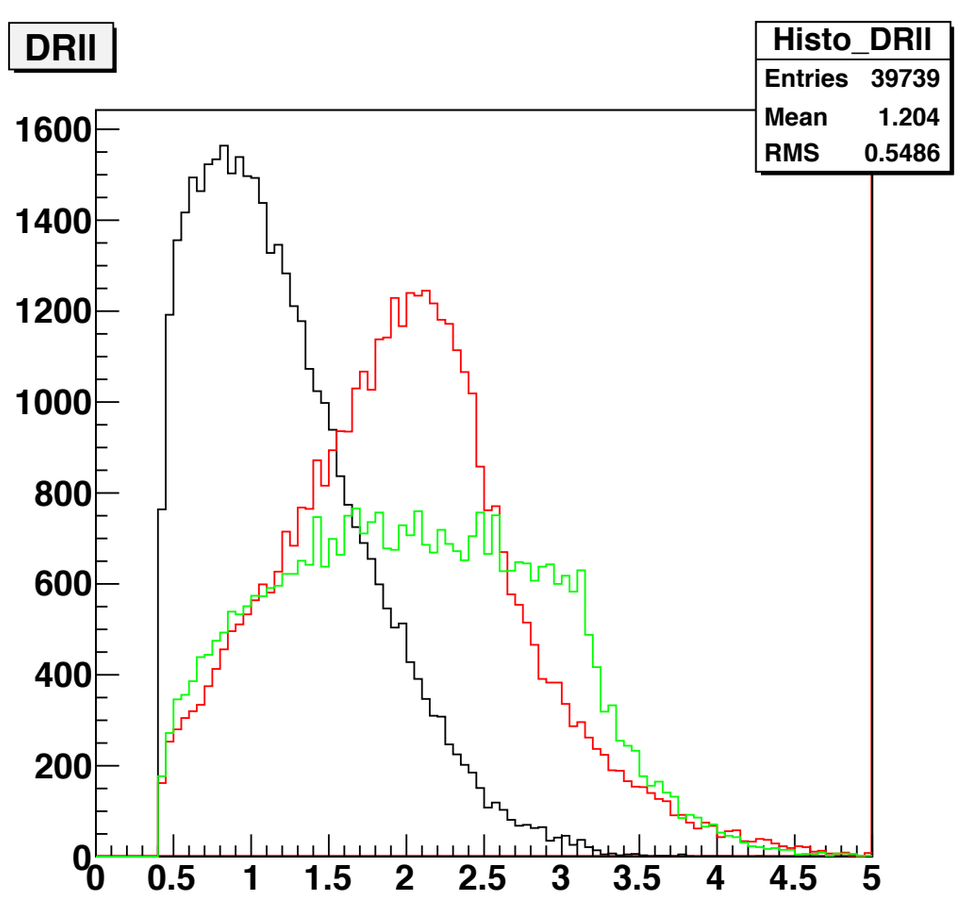
Also

```
if (ptl1 > 50 || ptl1 < 25) {
  continue;
}
if (ptl2 > 50 || ptl2 < 25) {
  continue;
}
```

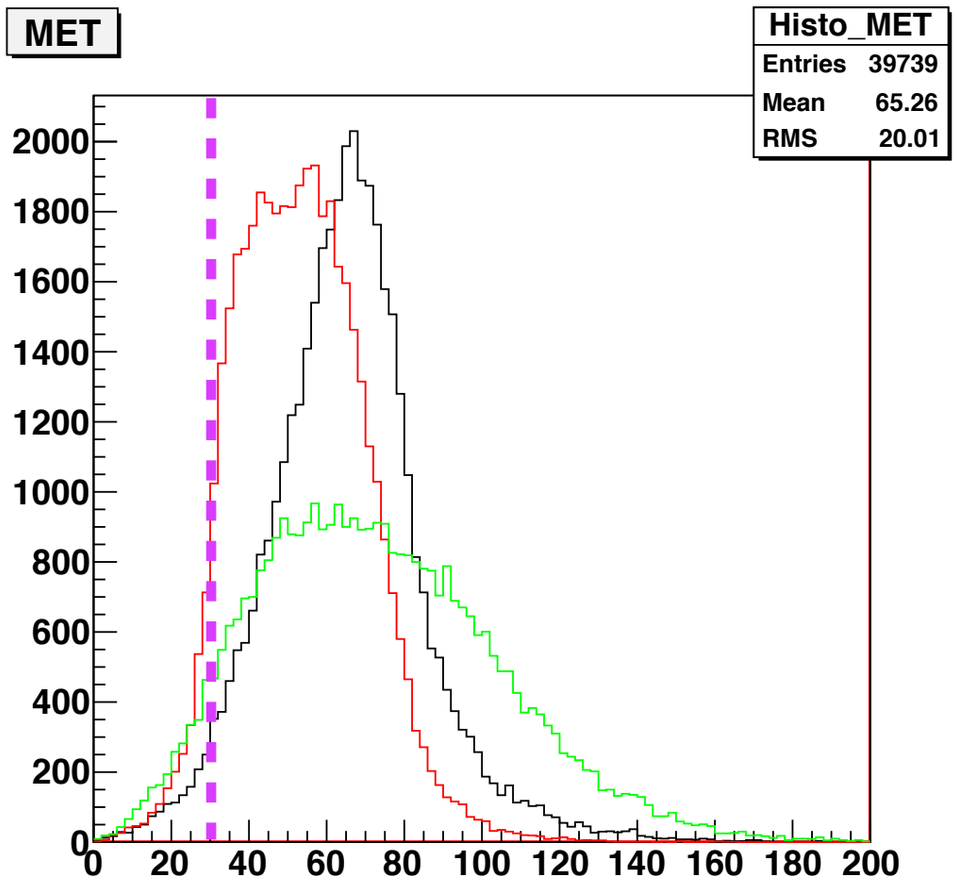
Signal Vs Background

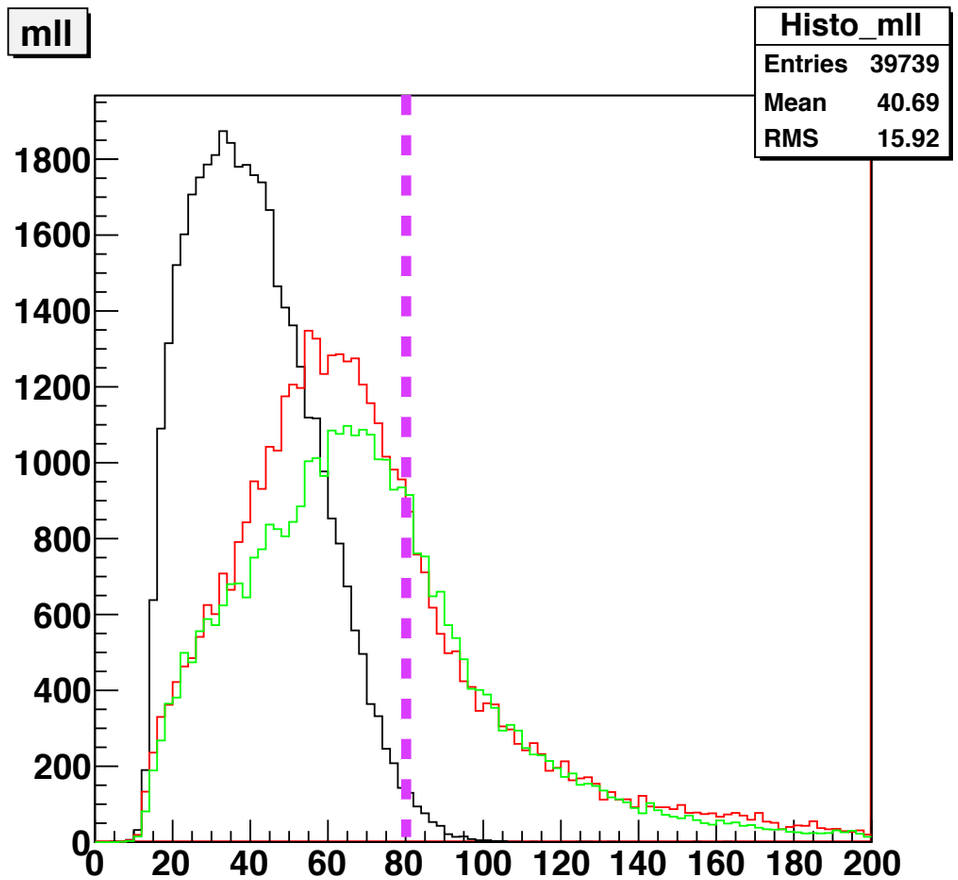


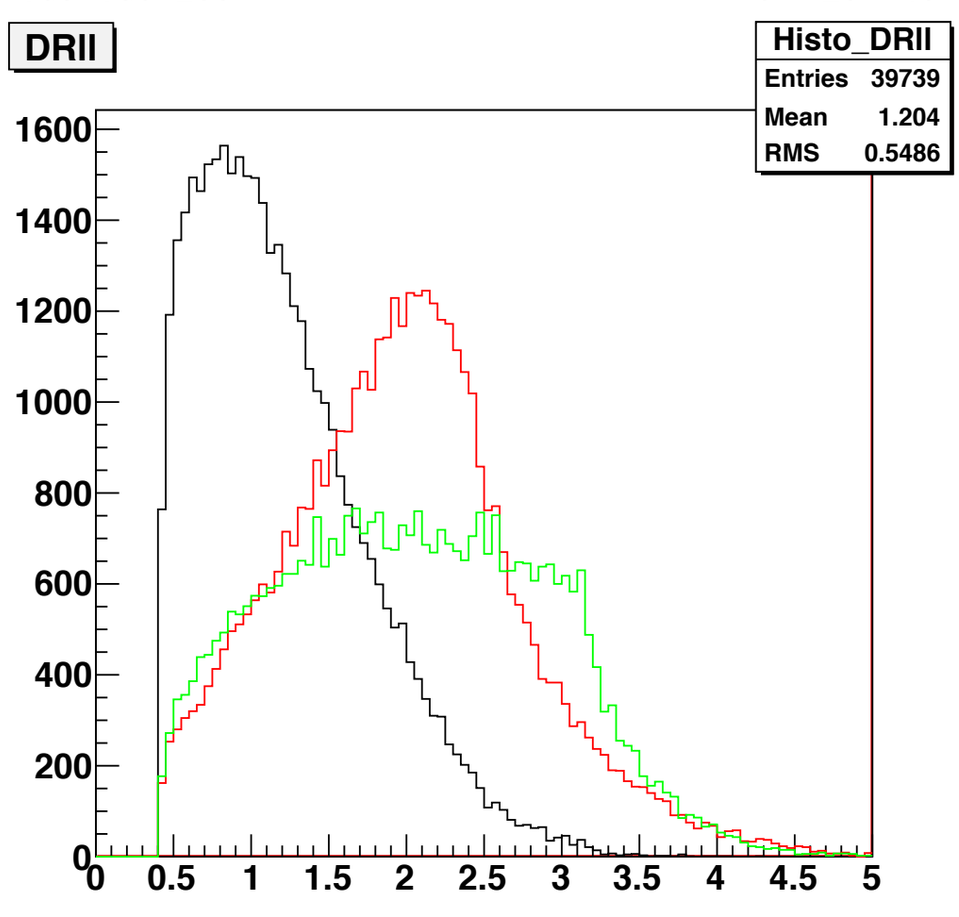




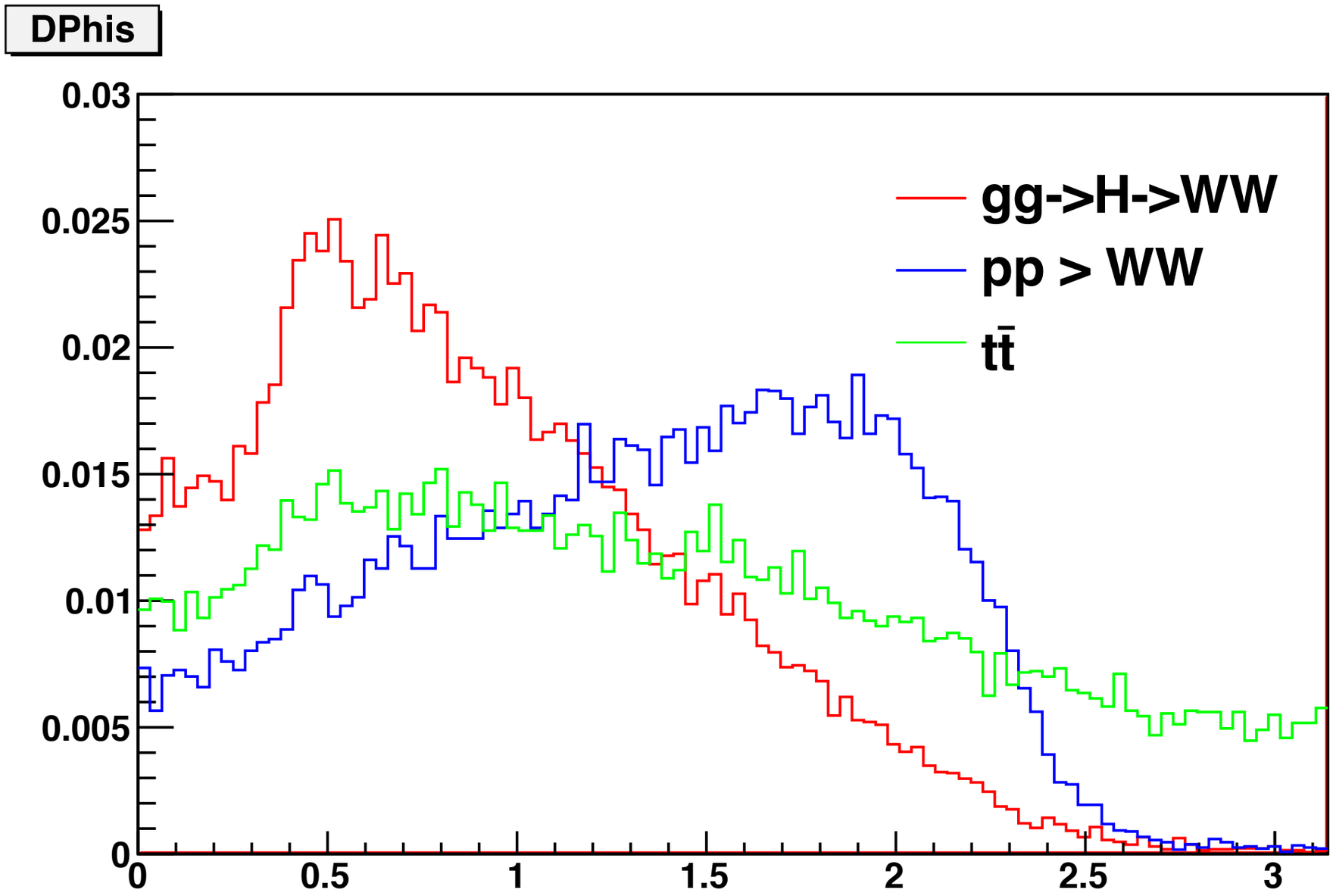
Signal Vs Background



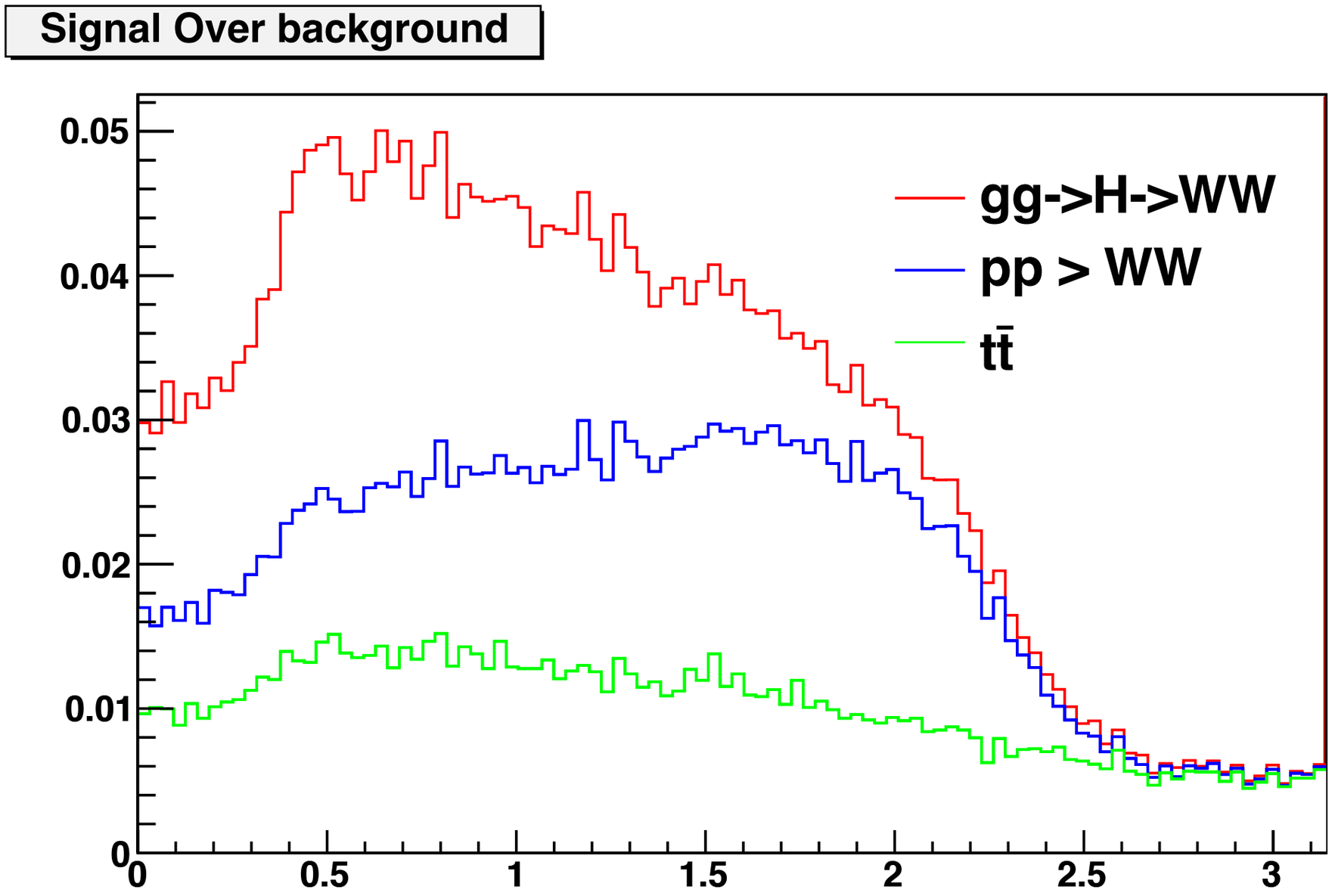




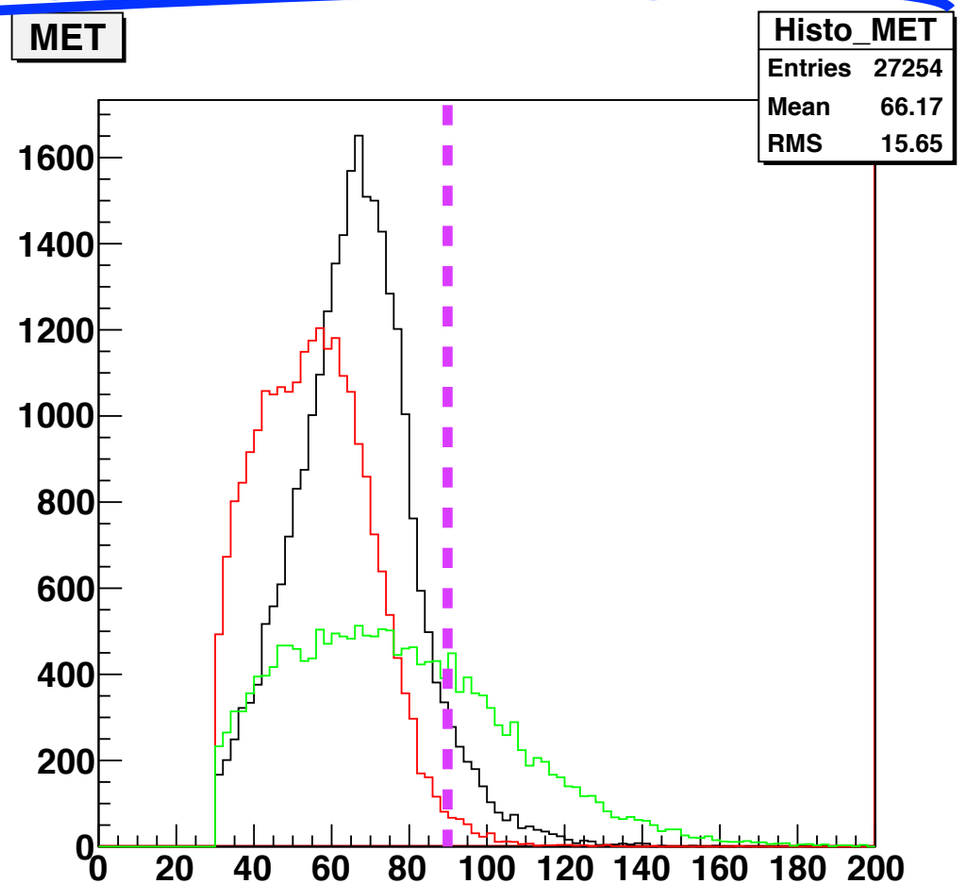
Signal Vs Background

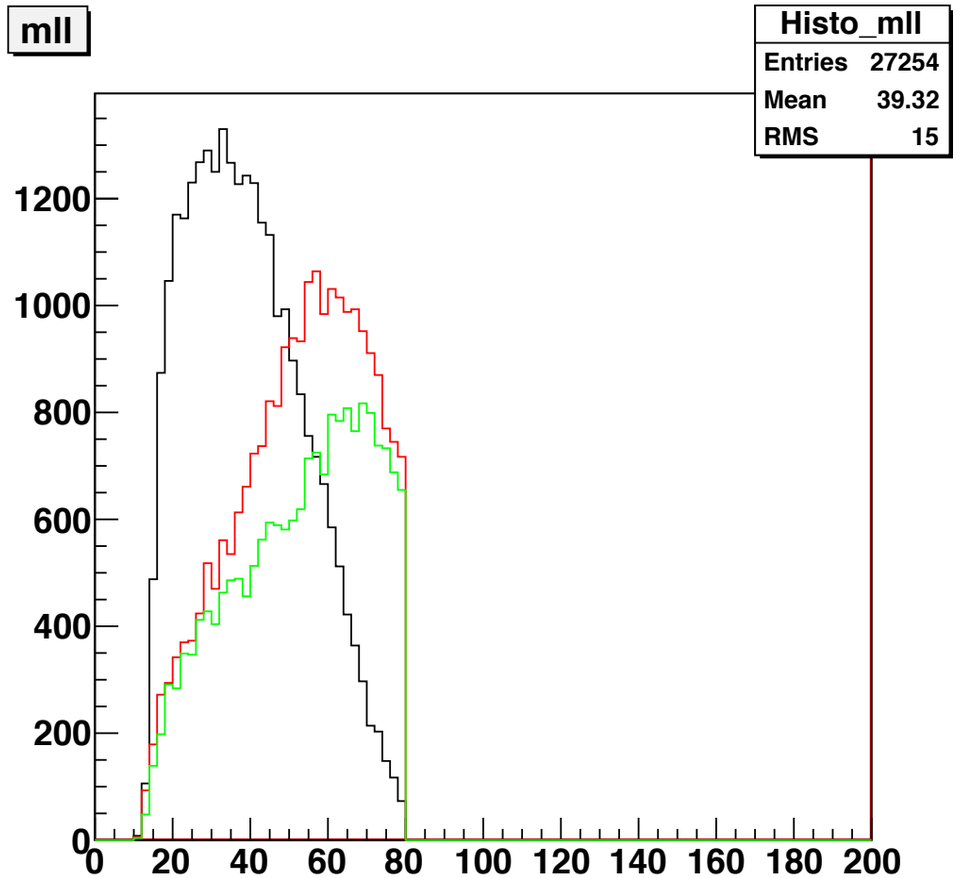


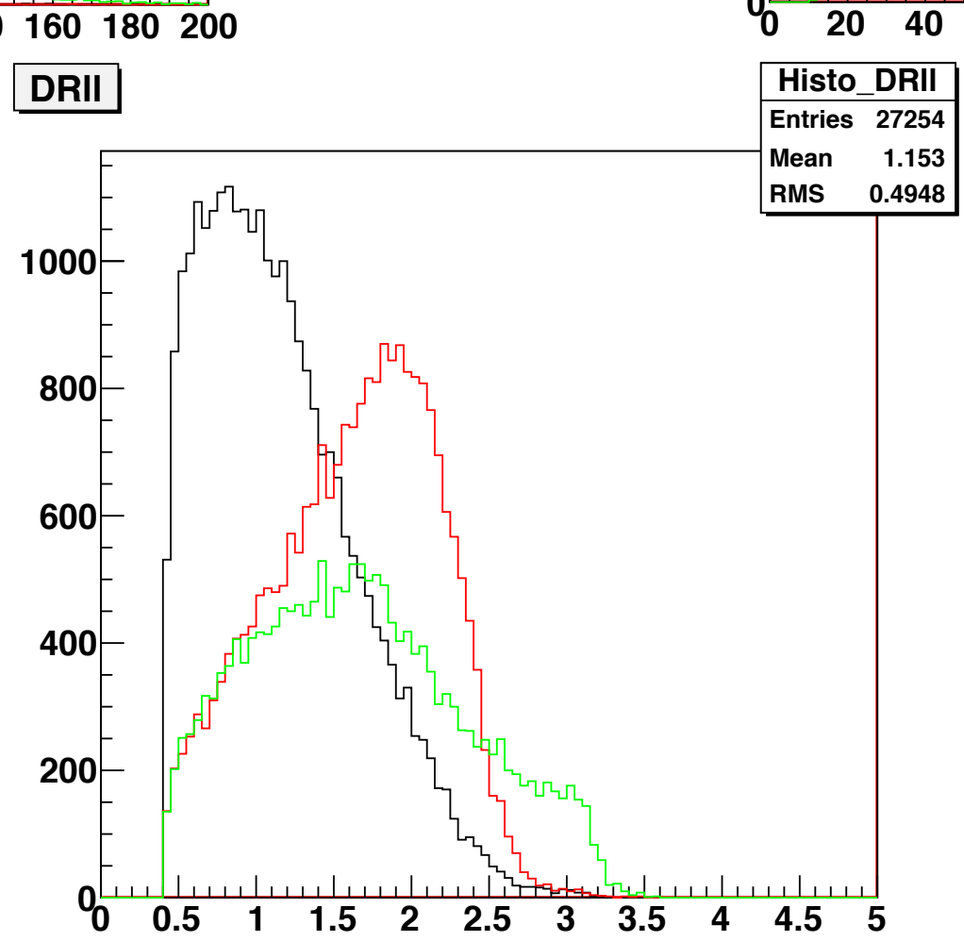
Signal Vs Background



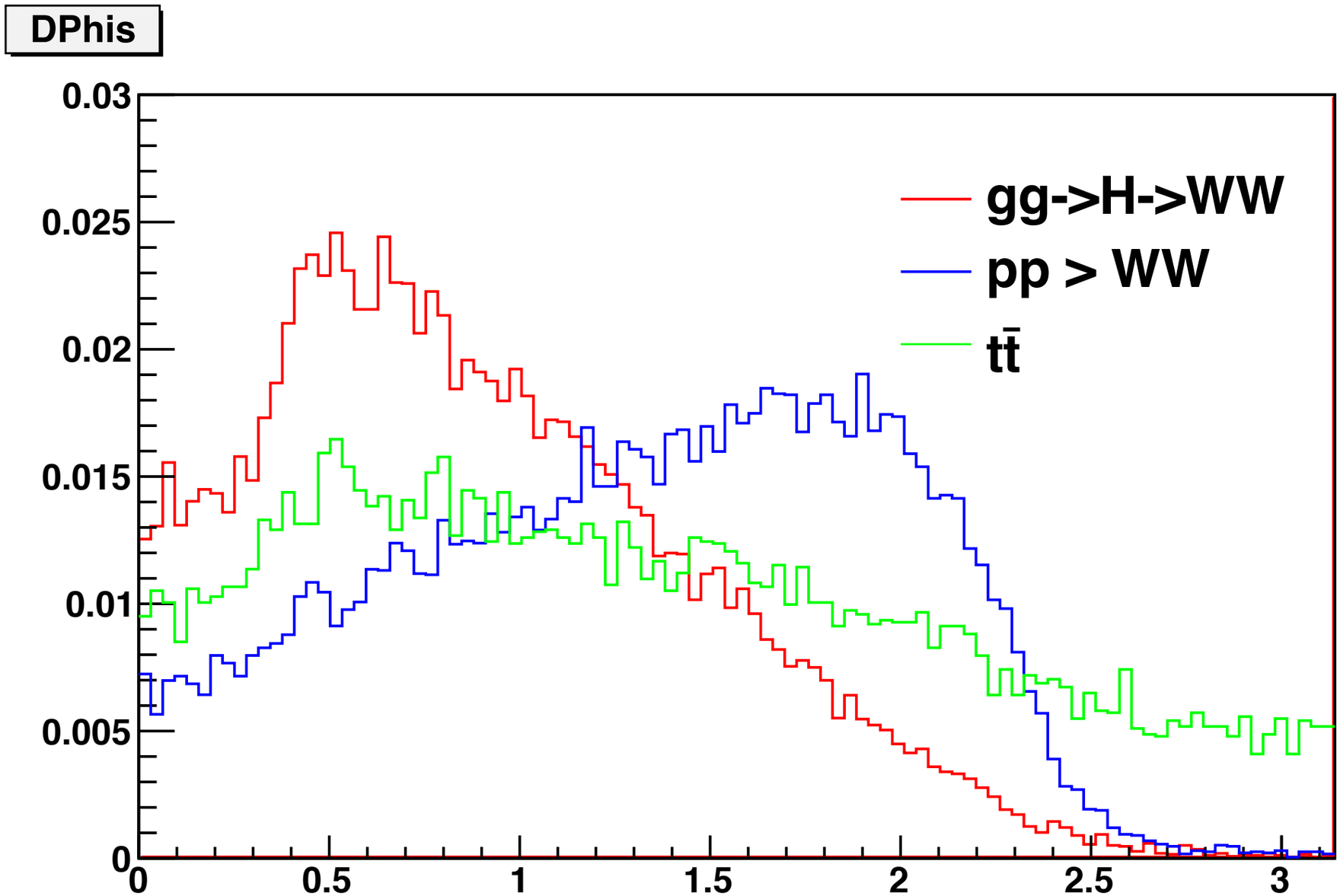
Signal Vs Background



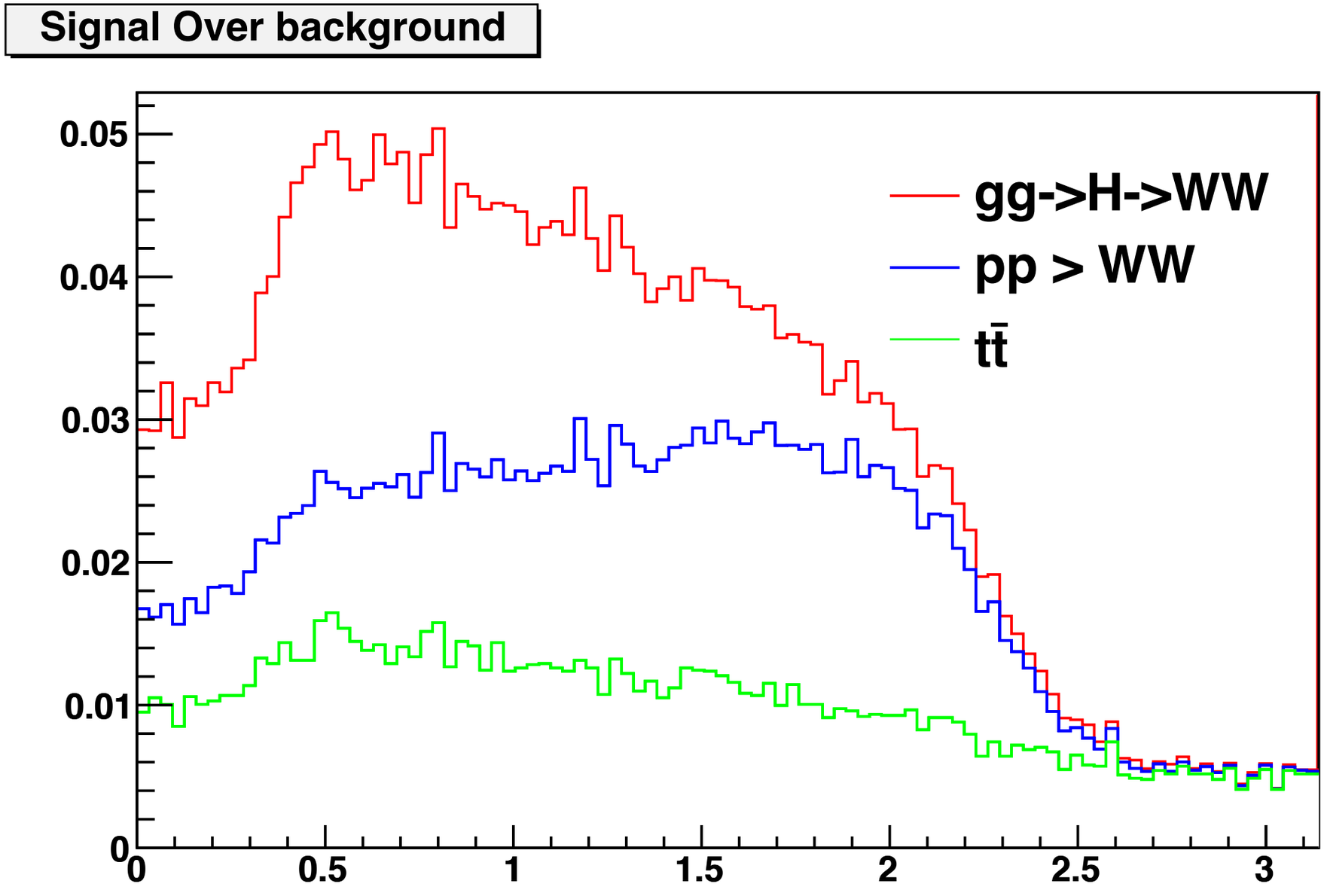




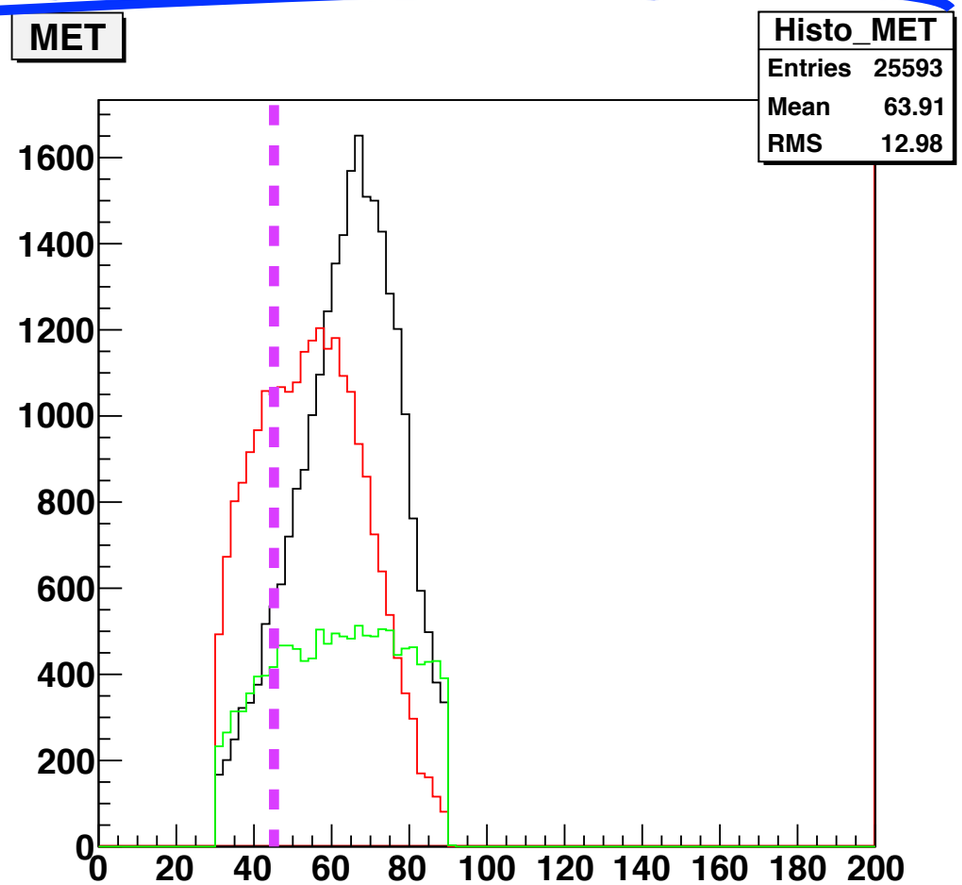
Signal Vs Background

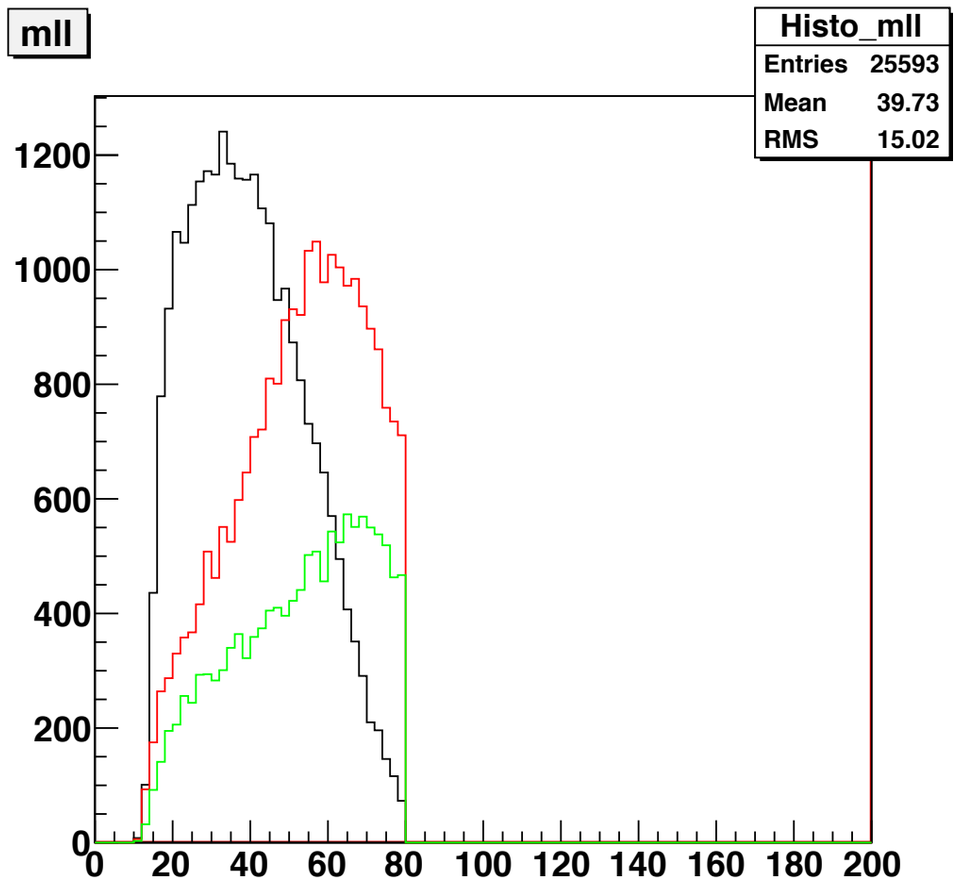


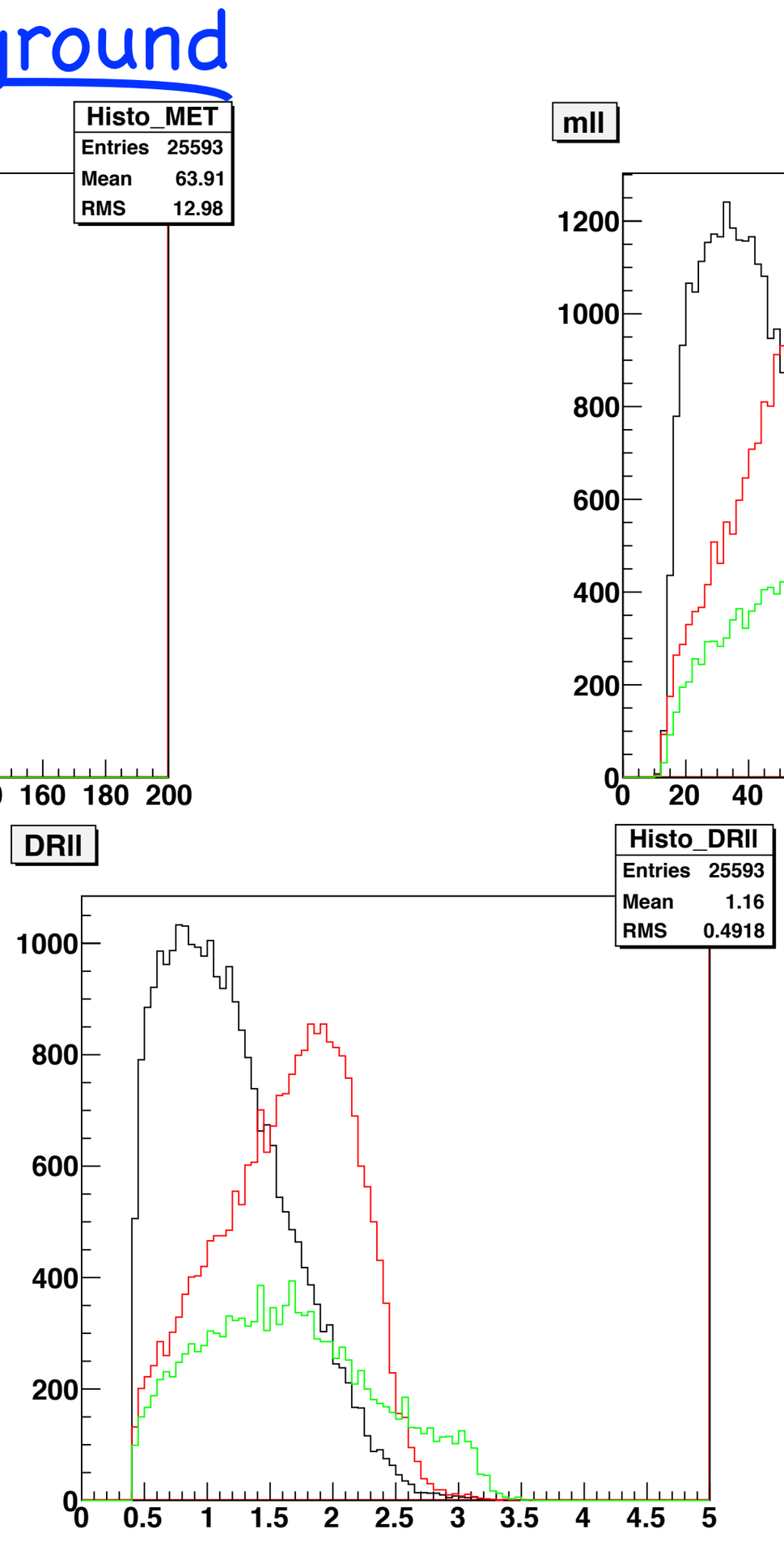
Signal Vs Background



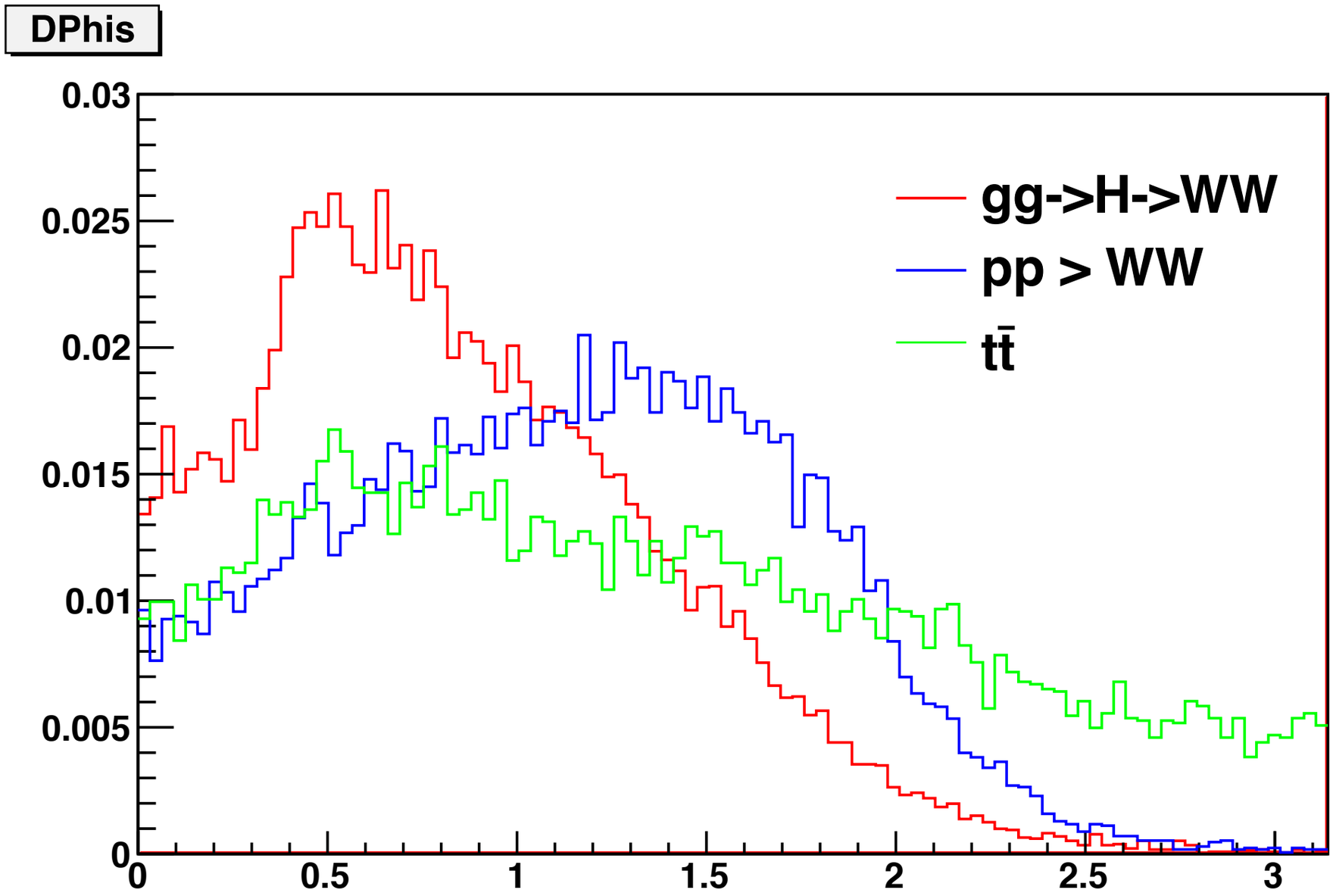
Signal Vs Background



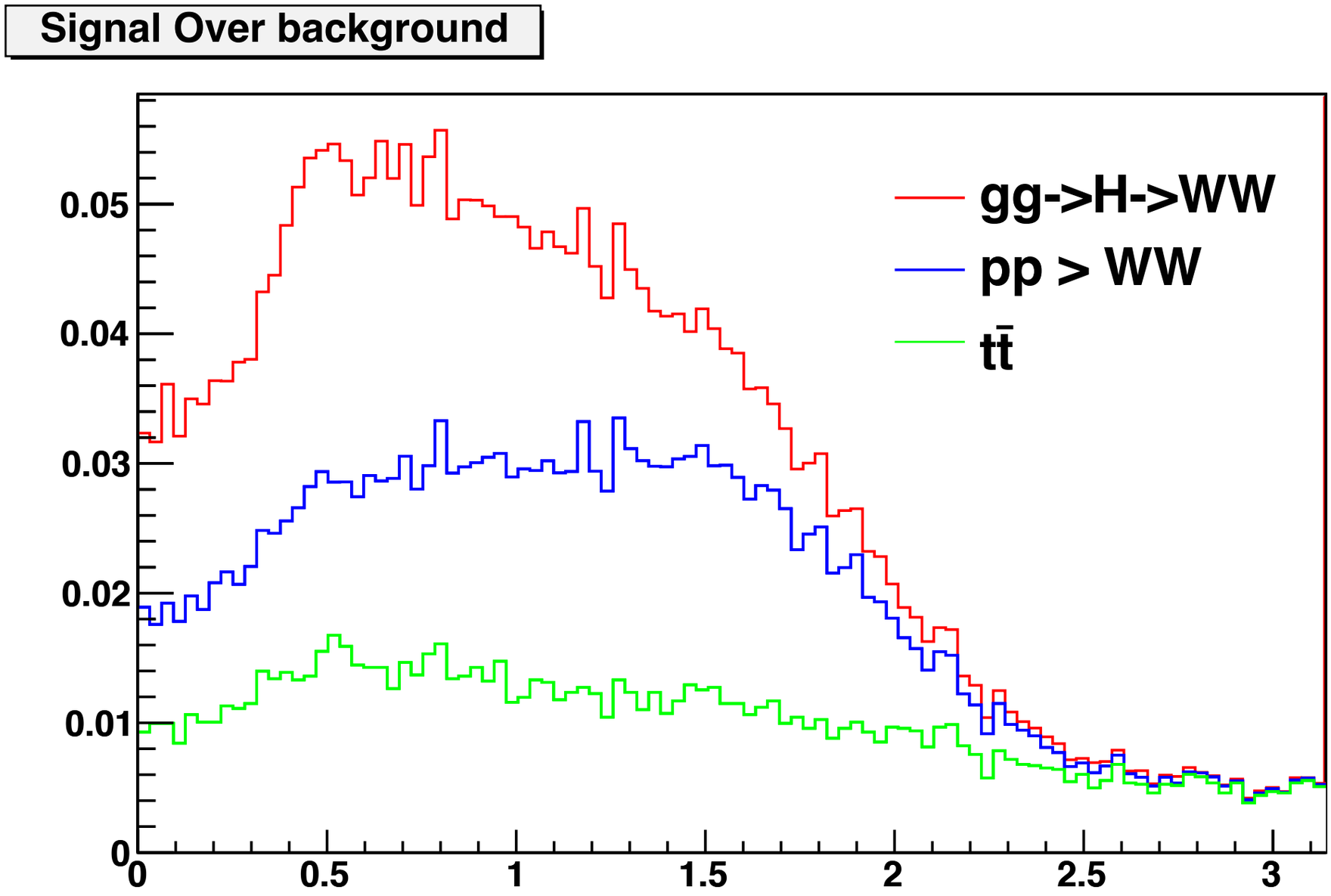




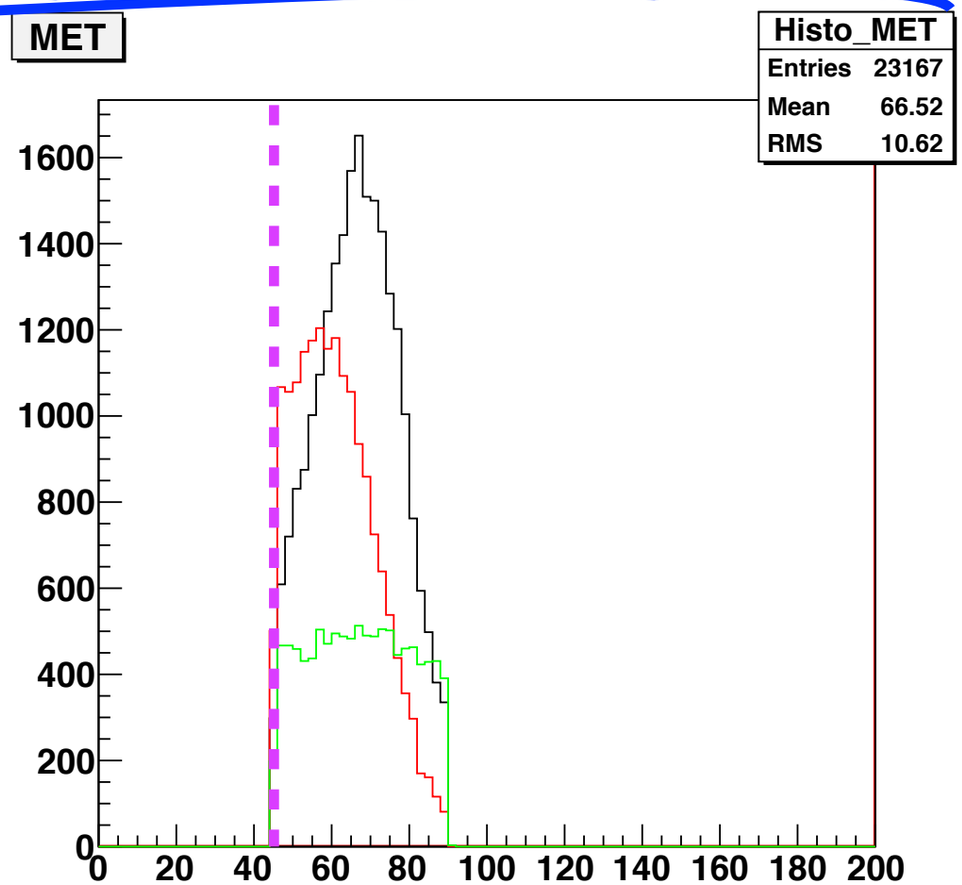
Signal Vs Background

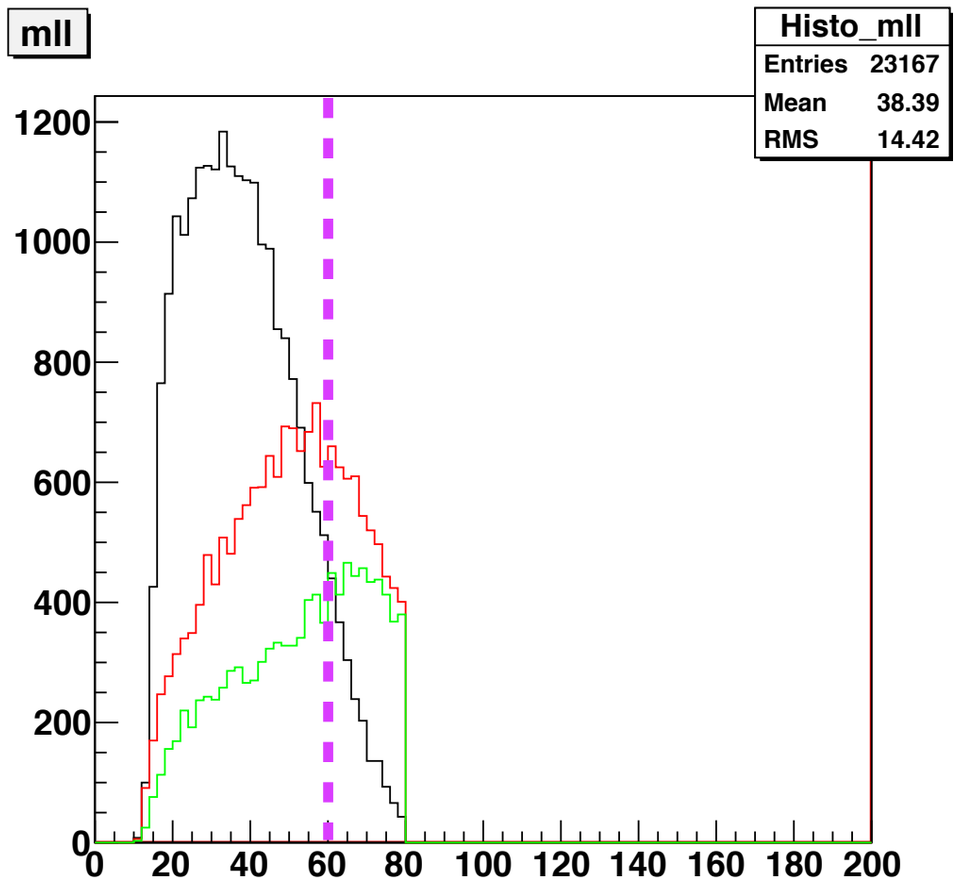


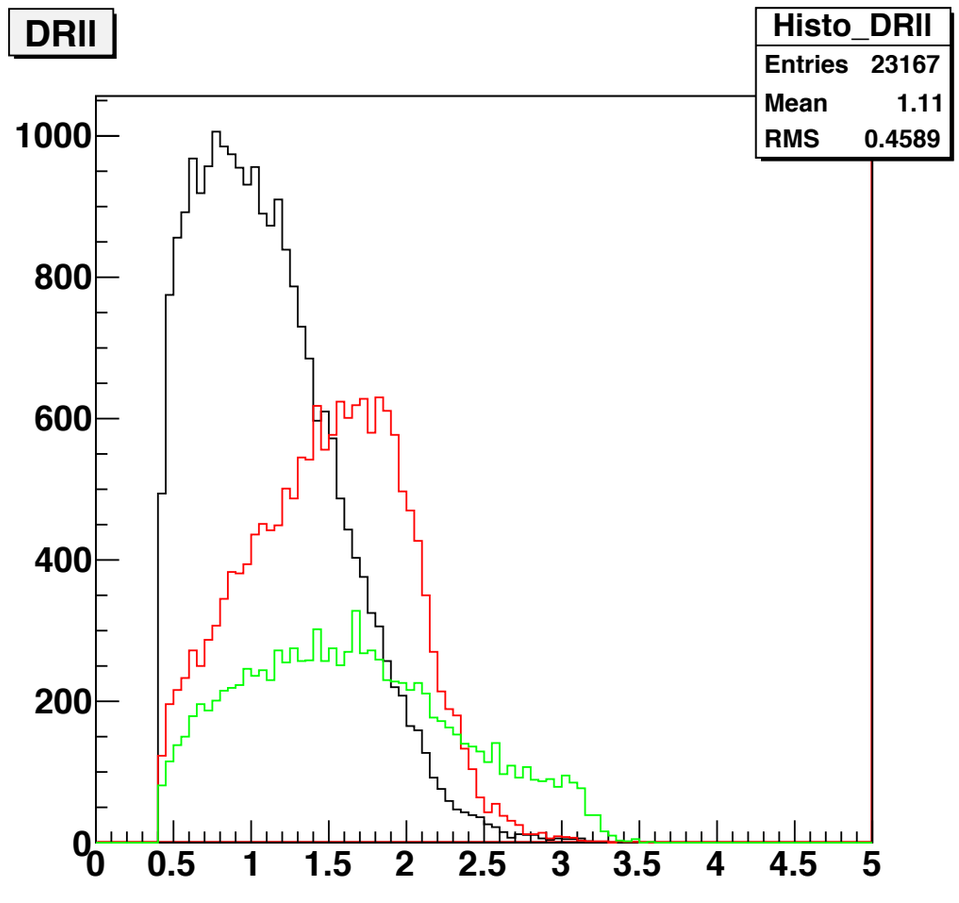
Signal Vs Background



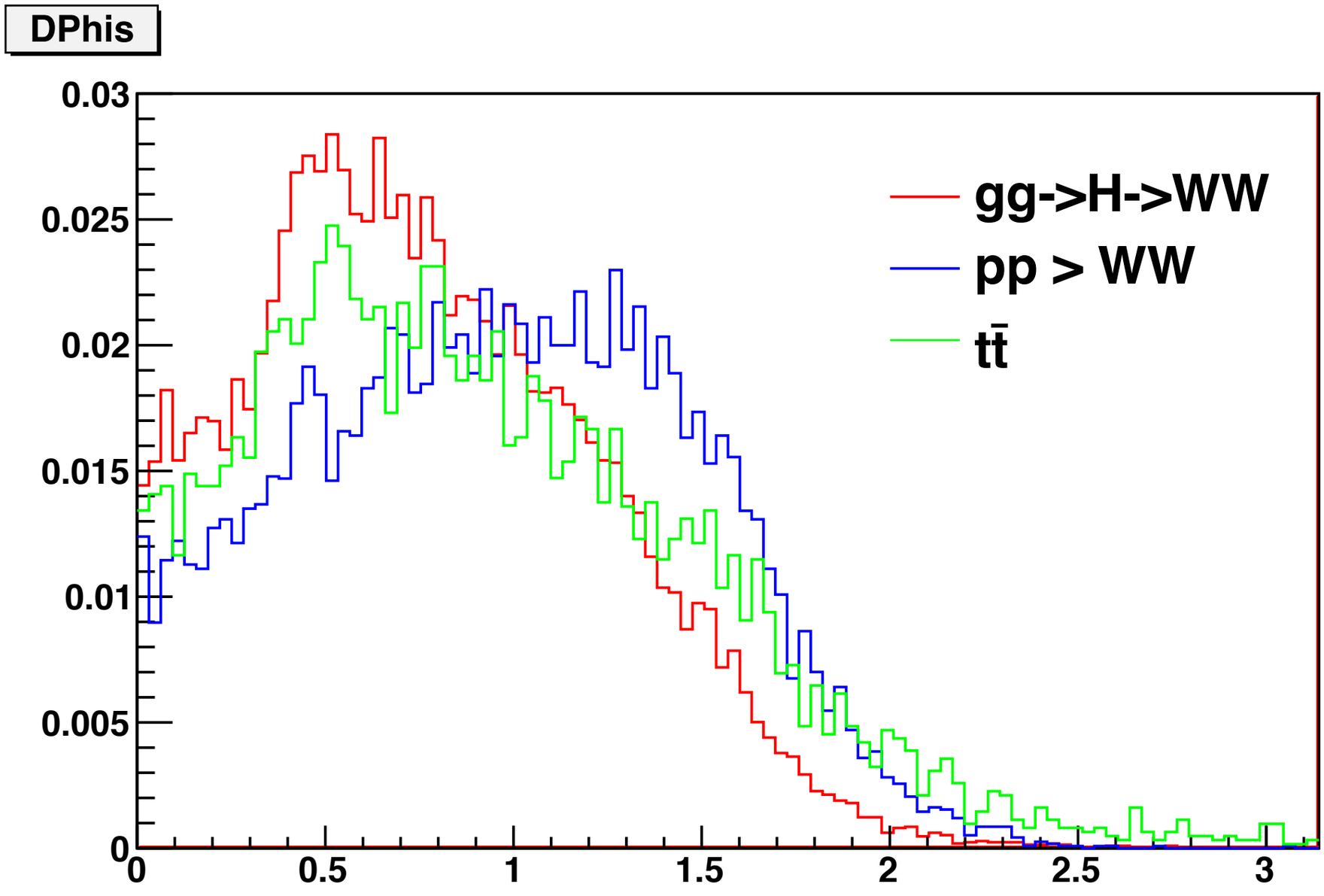
Signal Vs Background



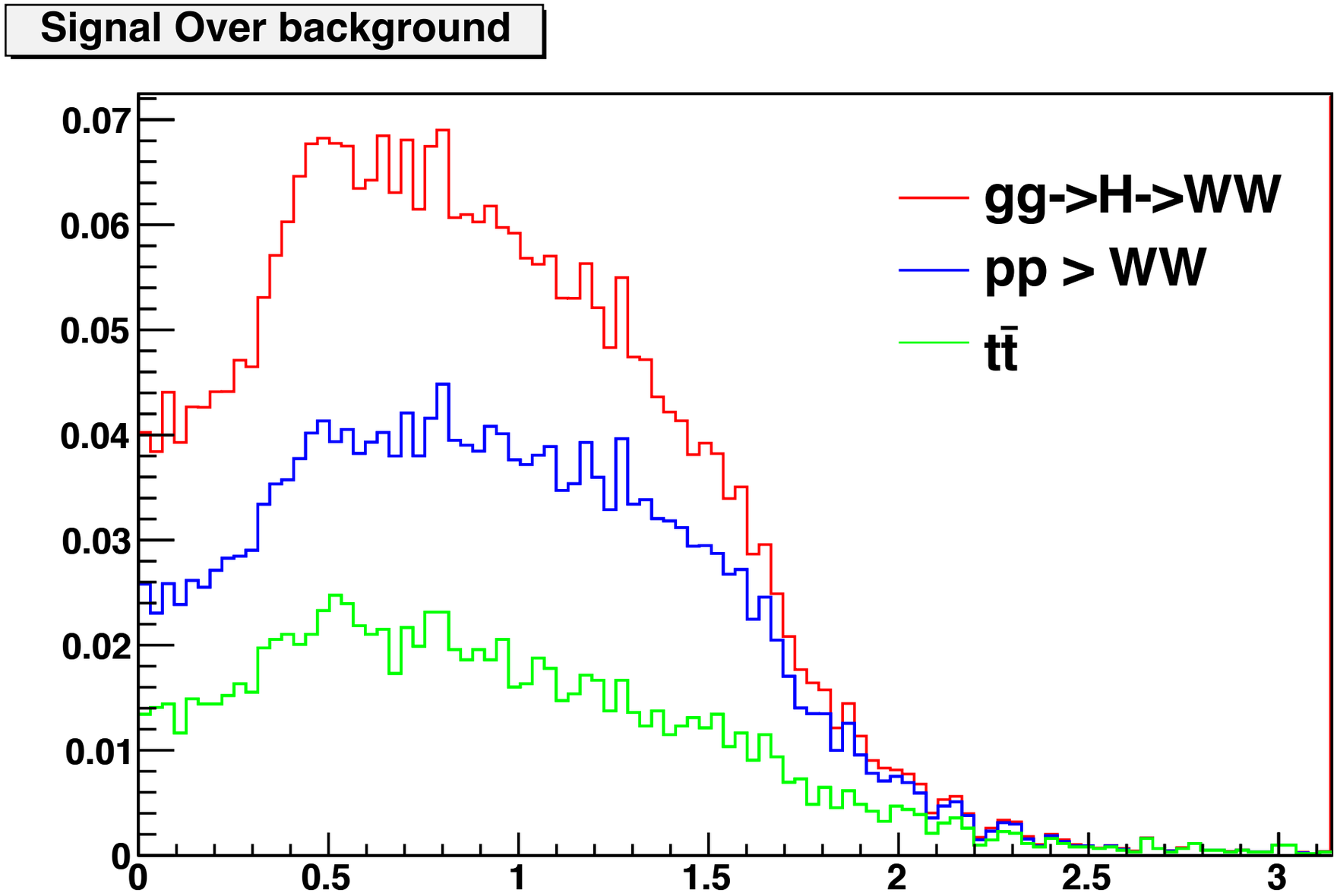




Signal Vs Background



Signal Vs Background



Signal Vs Background