ROOT 3

Gökhan Ünel / UCI
HPFBU-2012

Dinlenme gününden sonra root hatırlatması

içindekiler

- Neydi ki? (kısa kısa notlar)
  - ➡ buraya da bakın: http://root.cern.ch/root/soeren/
- bir kütük buldum!
  - ➡ histogramlar doldurmak, çizmek
  - ➡ ntuple nedir? nasıl bakılır? nasıl çizilir?
- eğri yakıştırmak
  - ➡ GUI vs komut satırı
  - ➡ sonuçları alıp nasıl kullanmalı?
- Betik (macro) yazmak
  - ➡ Bu işareti görünce betiğe yazılacağını anlayın:

- İşlevler (TF1)
- TGraphs

Neydi ki?

- Root açılış resmini görmemek için (eğer hemen olmuyorsa çok yavaştır. BG)
  - ➡ `root -l` (veya bir alias yaparsınız!)
- Root iyi bir hesap makinesidir
  - ➡ `root [0] sqrt(1.2)*2`
- Değişkenler programlanabilir
  - ➡ `root [0] float k1=sqrt(1.2)*2`
  - ➡ `root [1] cout << k1<<endl;`
  - ➡ `2.19089`

Kırmızı ile bilgisayarın yanıtları yazılı

- Kabuk komut satırında:
  - ➡ `root -l -q -x abc.C (hemen aç, abc.C yi çalıştır, bitince çık.)`
- hızlıca bir kütüğü yüklemek:
  - ➡ `root -l pp-DD-ZjWj/validation/events.root`
  - ➡ `Attaching file .//pp-DD-ZjWj/validation/events.root as _file0...`

Bir kütük buldum

- aynı kütüğü (ve ikincisini) ROOT içinden açarsak
  - `TFile *_file0 = TFile::Open("events1.root")`
  - `TFile *_file1 = TFile::Open("events2.root")`

- kütüğün içinde ne var?
  - `.ls`

```
TFile**         105200.root
 TFile*         105200.root
  KEY: TDirectoryFile    FF_1;1    FF_1
  KEY: TDirectoryFile    FF_1_01_jesn;1  FF_1_01_jesn
  KEY: TDirectoryFile    FF_1_01_jesp;1  FF_1_01_jesp
```

aynı unix ortamındaki gibi dizinler.

- dizinin içinde ne var?

```
FF_1->cd()
.ls
KEY: TParameter<double> TRGm;1    Named templated parameter type
KEY: TH1F    eff;1selection efficiencies
KEY: TNtuple rntuple;1 run info
KEY: TH1F    metx0;1   met x (GeV)
KEY: TH2F    hmDD0b0;1 M_{D_{Wj1}} vs M_{D_{Wj2}} 0b
```

Neymiş bunlar?

➡ KEY: TParameter<double> TRGm;1 Named templated parameter type

- parametre değerini okumak:

➡ root [3] TRGm->GetVal()

➡ (const double)2.00000000000000000e+00

- ödev: TParameter değeri nasıl verilir?

adı TGm olan bir float değişken.

➡ KEY: TH1F eff;1selection efficiencies

➡ KEY: TNtuple rntuple;1 run info

1 boyutlu bir histogram, adı eff açıklaması ; den sonra verilmiş.

bir ntuple, adı `rntuple`, ayrıntılar daha sonra!

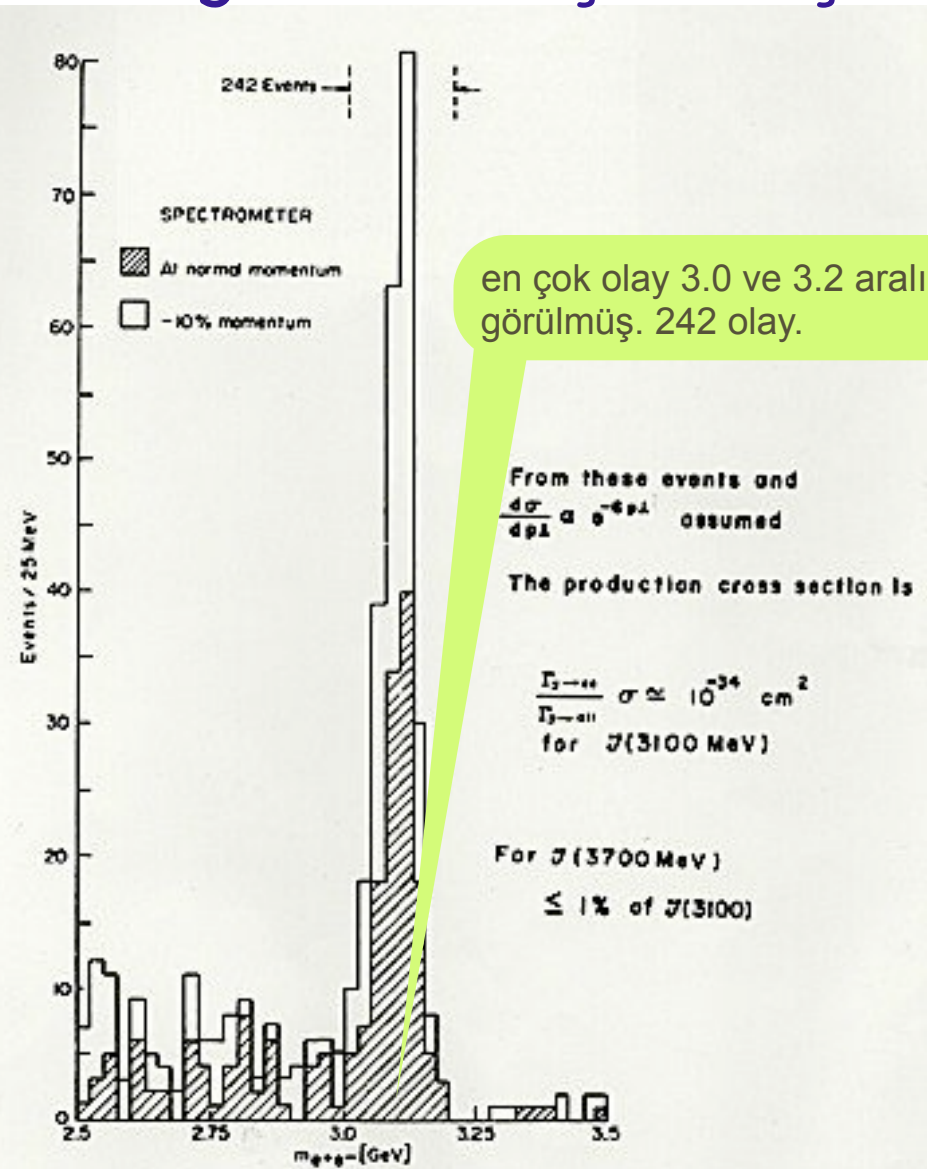
➡ KEY: TH1F metx0;1 met x (GeV)

➡ KEY: TH2F hmDD0b0;1 M_{D_{Wj1}} vs M_{D_{Wj2}} 0b

2 boyutlu bir histogram, adı `hmDD0b0`, açıklamadaki latex tarzı yazıma dikkat !

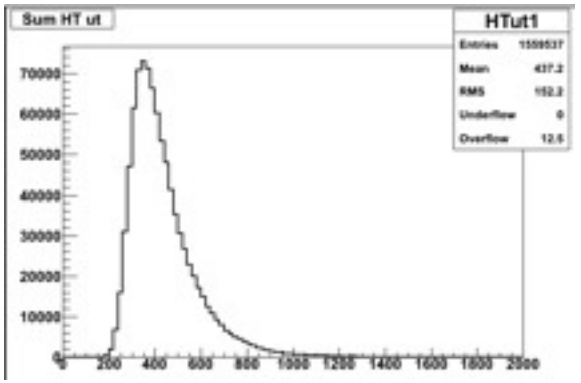
TH1 1

- TH1F: 1 boyutlu, içinde gerçek sayıların olduğu histogram. değerlerin kaç defa çıktığını veya olay adedini gösterir.

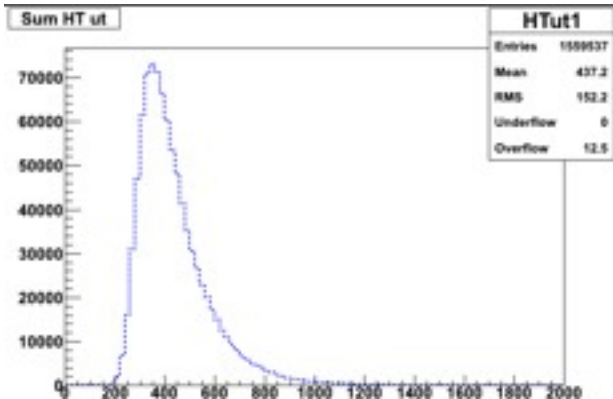


en çok olay 3.0 ve 3.2 aralığında görülmüş. 242 olay.

- Bir histogram çizelim görelim:
  - ➡ `HTut1->Draw()`

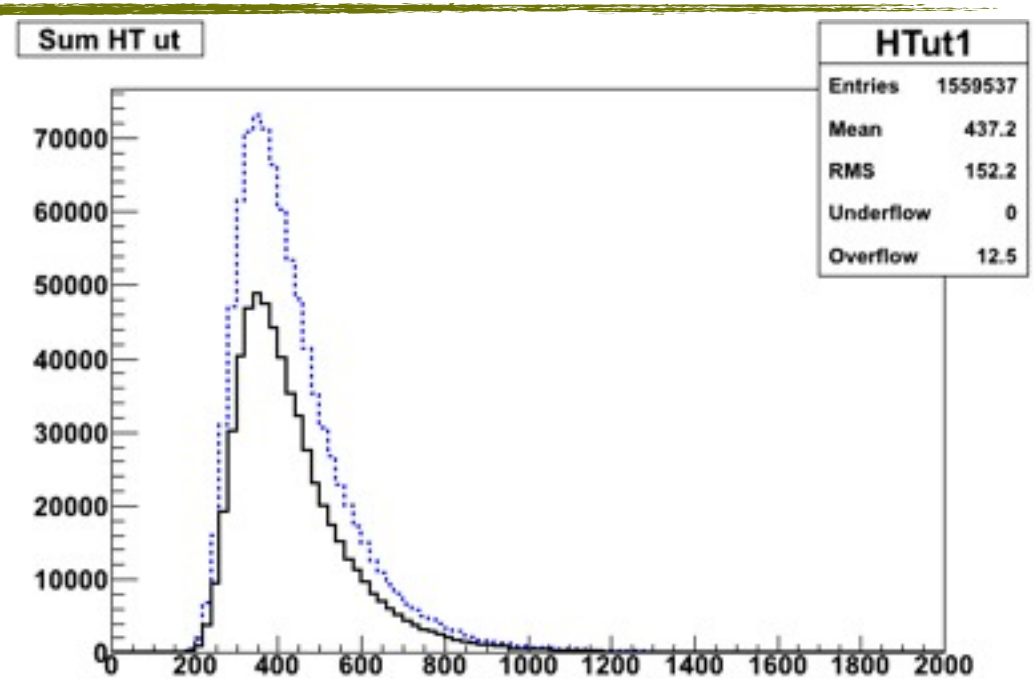


- Rengini vs değişmek için
  - ➡ `HTut1->SetLineColor(kBlue)`
  - ➡ `HTut1->SetLineStyle(2)`
  - ➡ `HTut1->Draw()`



TH1 2

- Başka bir histogramı üstüne çizdirmek için:
  - ➡ `HT1b1->Draw("same")`
    - ödev: çizimin üzerine çift tıklayınca çıkan menü ile oynayın. Özellikle de DrawPanel ile. Farklı çizim usullerini deneyin. Kafes (grid) koyun.



- bunları bir betik içinde yapın
  - bir metin düzenleyicide yazın =>
  - ➡ `root -l -x dene1.C`

```
{
TFile *_file0 = TFile::Open("105200.root");
FF_1->cd();
HTut1->SetLineColor(kBlue);
HTut1->SetLineStyle(2);
HTut1->Draw();
HT1b1->Draw("same");
}
```

- Belli bir oranda çizmek istesek?
  - ➡ `TCanvas *c1= new TCanvas(`
    - ayraç'a kadar yazın ve TAB'a basın
  - ➡ `TCanvas TCanvas(Bool_t build = kTRUE)`
  - ➡ `TCanvas TCanvas(const char* name, const char* title = "", Int_t form = 1)`
  - ➡ `TCanvas TCanvas(const char* name, const char* title, Int_t ww, Int_t wh)`

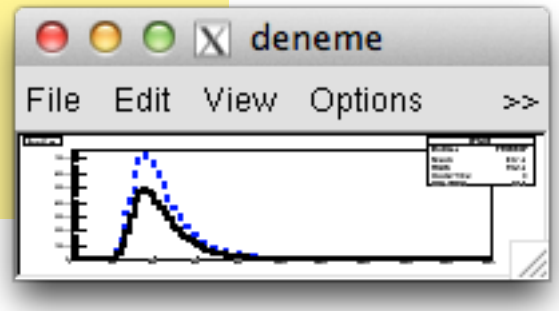
kısa ve kalın yapmak için

```
TCanvas *c1= new TCanvas("c1", "deneme", 200, 80);
c1->cd();
```

TH1 3

- betiği yenileyelim
- deneyelim
  - ➡ `root -l -x dene1.C`
- Biraz küçük oldu!!

```
{

TFile *_file0 = TFile::Open("105200.root");
FF_1->cd();
HTut1->SetLineColor(kBlue);
HTut1->SetLineStyle(2);

TCanvas *c1= new TCanvas("c1 ", "deneme", 200, 80);
c1->cd();

HTut1->Draw();
HT1b1->Draw("same");

}
```



- Pencerenin X ve Y boylarını kullanıcı versin. Betik başına ekleyelim:

x=400 ve y=160 öntanımlı olsun

  - ✍ `int dene1 (int xw=400,int yw=160)`
    - Dikkat: işlev adı betik adıyla aynı
    - C++ işlev programlama yapıyoruz. bu yeni değişkenleri kullanalım.
    - işlevimiz şimdilik 0 döndürsün.
    - çalıştıralım...
  - ➡ `root -l -x "dene1.C(100,200)"`

```
int dene1 (int xw=400, int yw=160)
{

TFile *_file0 = TFile::Open("105200.root");
FF_1->cd();
HTut1->SetLineColor(kBlue);
HTut1->SetLineStyle(2);

TCanvas *c1= new TCanvas("c1 ", "deneme", xw, yw);
c1->cd();

HTut1->Draw();
HT1b1->Draw("same");

return 0;
}
```

TH1 4

- Çizimi saklamak için son komuttan önce betiğe ekleyin:
  - ✍c1->SaveAs("dene1.pdf");
  - ✍c1->SaveAs("dene1.jpg");
    - deneyelim:

```
➡root -l -q -x "dene1.C(100,200)"
➡root [0]
➡Processing dene1.C(100,200)...
➡Info in <TCanvas::Print>: pdf file dene1.pdf has been created
➡Info in <TCanvas::Print>: file dene1.jpg has been created
➡(int)0
```

```
new-host-2:161 ngu$ ls  dene1*
dene1.C         dene1.jpg       dene1.pdf
```

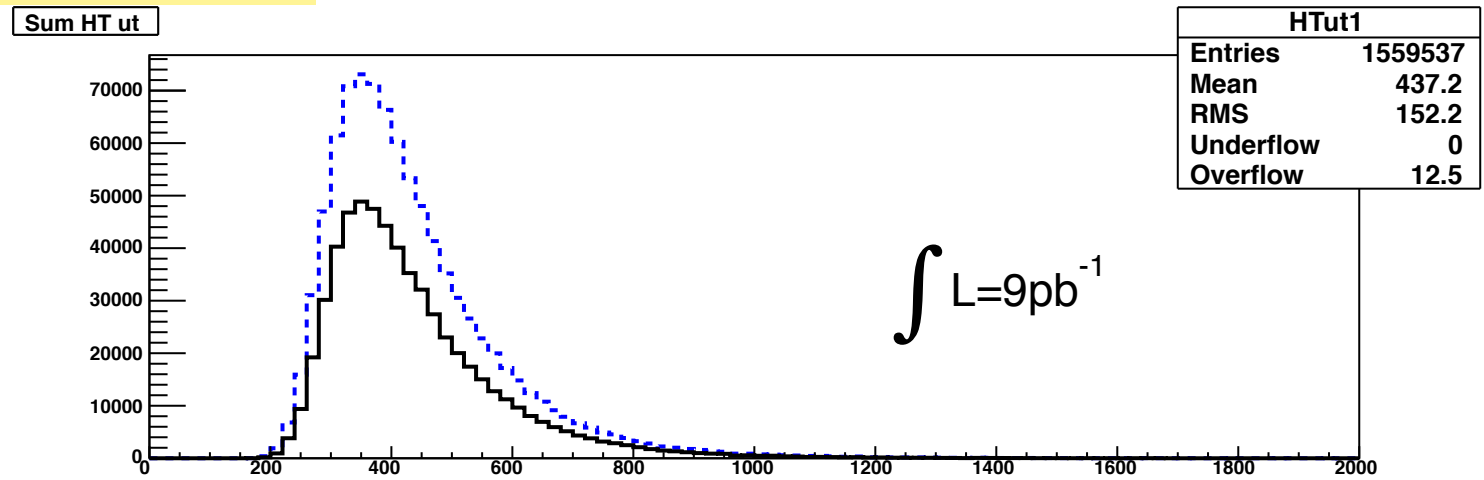
- Betiğin son hali yanda =======>

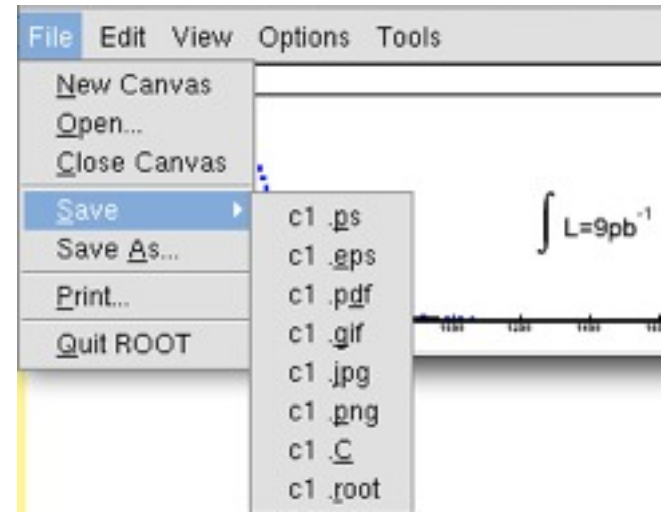
```
int dene1 (int xw=400, int yw=160)
{

TFile *_file0 = TFile::Open("105200.root");
FF_1->cd();
HTut1->SetLineColor(kBlue);
HTut1->SetLineStyle(2);

TCanvas *c1= new TCanvas("c1 ", "deneme", xw, yw);
c1->cd();

HTut1->Draw();
HT1b1->Draw("same");

c1->SaveAs("dene1.pdf");
c1->SaveAs("dene1.jpg");
return 0;
}
```

- Toplam ışınlık ekleyin $\int L=9pb^{-1}$
  - ✍TLatex lLumi;
  - ✍lLumi.SetNDC(1);
  - ✍TString TsLumi="#int L=9pb^{-1}";
  - ✍lLumi.SetTextFont(42);
  - ✍lLumi.SetTextSize(0.1);
  - ✍lLumi.DrawLatex(0.6,0.4,TsLumi);

özetlersek...

```
int dene1 (int xw=400, int yw=160)
{

TFile *_file0 = TFile::Open("105200.root");
FF_1->cd();
HTut1->SetLineColor(kBlue);
HTut1->SetLineStyle(2);

TCanvas *c1= new TCanvas("c1 ", "deneme", xw, yw);
c1->cd();

HTut1->Draw();
HT1b1->Draw("same");

TLatex lLumi;
lLumi.SetNDC(1);
TString TsLumi="#int L=9pb^{-1}";
lLumi.SetTextFont(42);
lLumi.SetTextSize(0.1);
lLumi.DrawLatex(0.6,0.4,TsLumi);

c1->SaveAs("dene1.pdf");
c1->SaveAs("dene1.jpg");
return 0;
}
```





- Tüm şeçenekler vs için root el kitabına bakın
- Genel olarak bir özelliği (renk, açı, boy...) değiştirmek için istediğiniz şekle sağ tıklayıp menu'den istediğiniz duruma getirin, sonra da macro (C) olarak saklayın. Bu C kütüğüne bakıp istediğiniz değişikliğin nasıl yapıldığını öğrenebilirsiniz...
  - ödev: 'Rebin' komutu nedir ?

TH1 5

- Başka TH1 işlemlerine örnekler
  - bir Histogramın eşini yapmak, böylece asıl histogram bozulmamış olur.

```
TH1* toplams = (TH1*) sih->Clone();
TH1* toplamq = (TH1*) dth->Clone();
```

  - bir Histogramın boyunu ayarlamak

```
toplams->Scale(6.8);
```

  - 2 Histogramı toplamak

```
toplamq->Add(toplams,toplamb);   // signal +bg = yalancı data
```

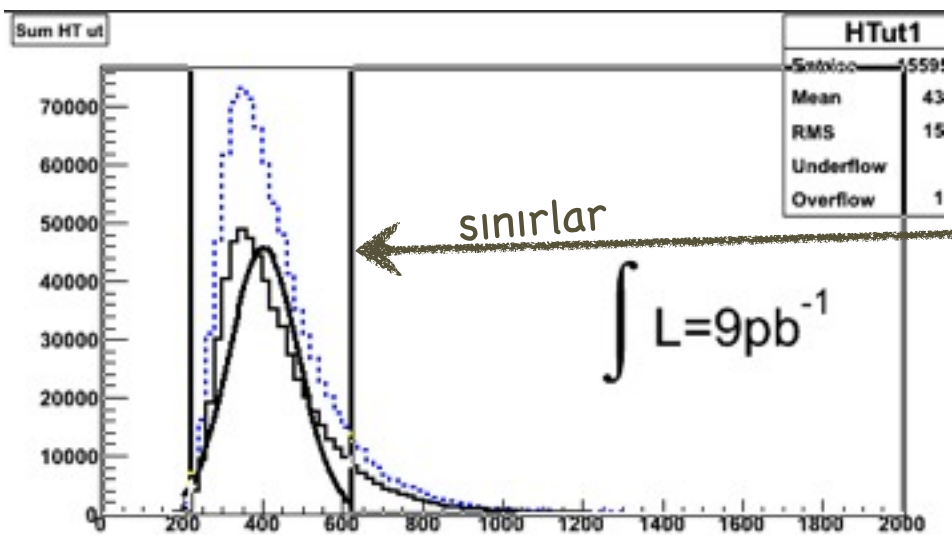
  - 2 Histogramı çarpmak

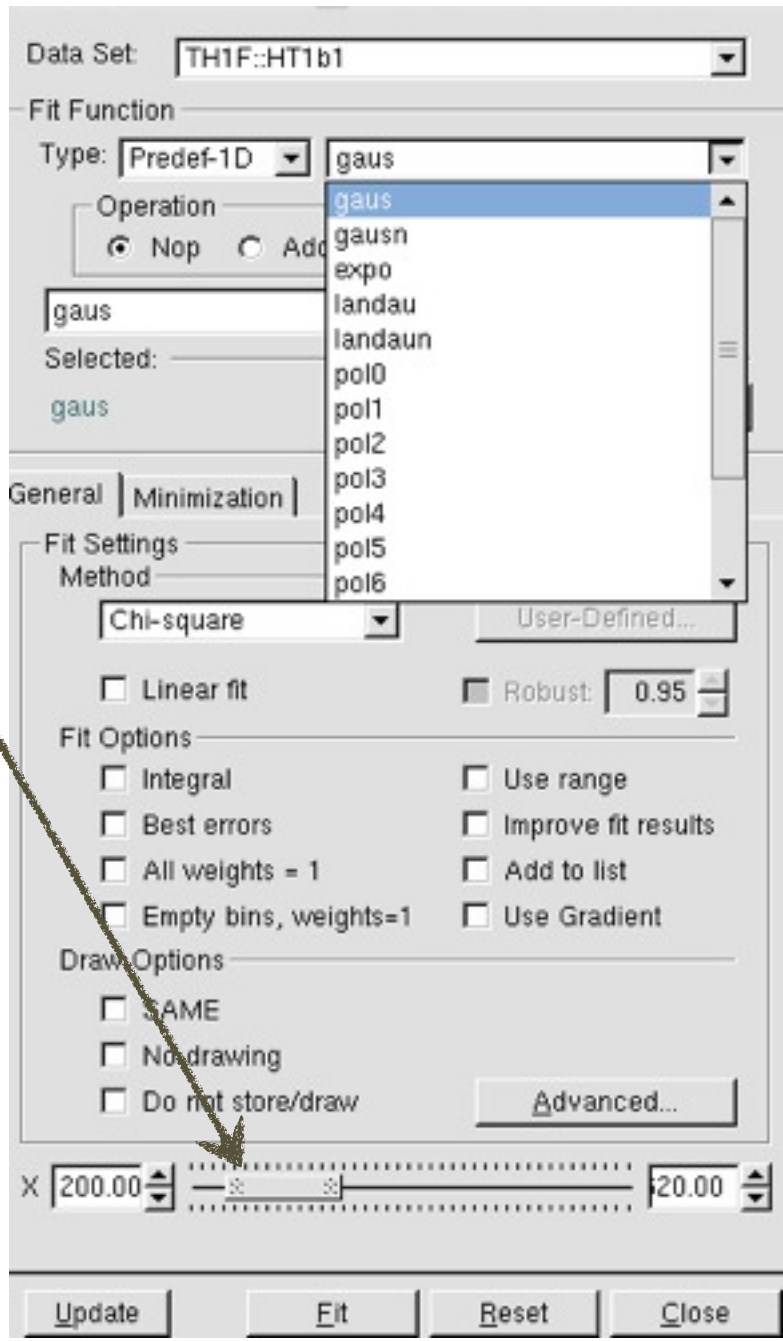
```
Multiply(const TH1* h1, const TH1* h2, Double_t c1 = 1, Double_t
  c2 = 1, Option_t* option = "")
```

- TH1 Hakkında her bilgiyi burada bulabilirsiniz:
  - http://root.cern.ch/root/html/TH1.html

Eğri yakıştırmak 1

- En basit yolu GUI kullanarak
  - Histograma sağ tıklayın, menu'den FitPanel'i seçin
  - bazı işlevler öntanımlı (gaus, landau, polynom...)
  - yakıştırma özellikleri seçilebilir.
    - örnek: yakıştırılırken kullanılacak bölge
  - Fit düğmesine basın. Sonucu alın:



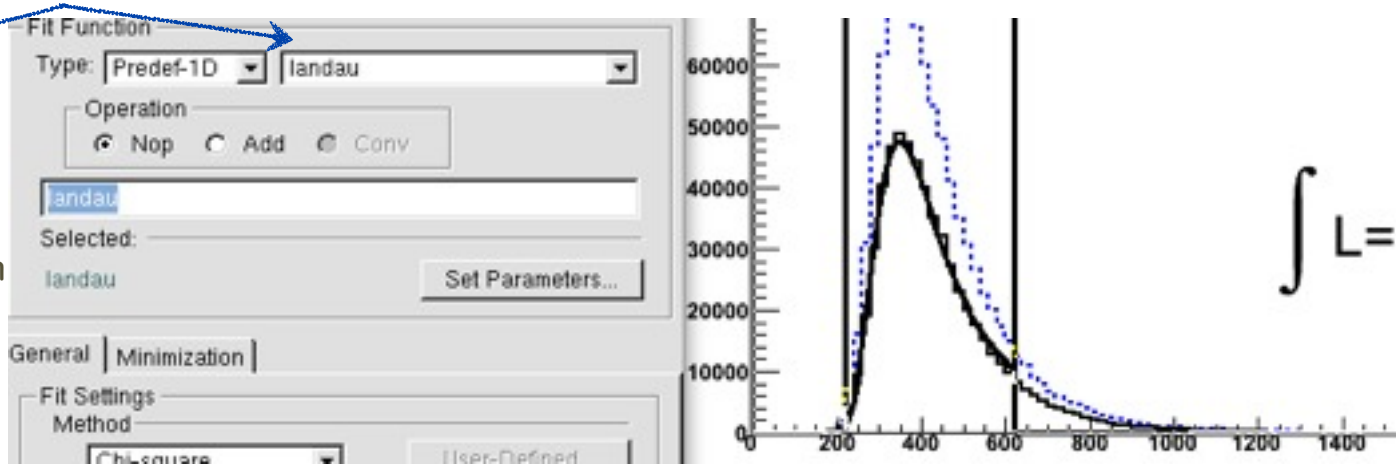



```
root [1]  FCN=48571.3 FROM MIGRAD    STATUS=CONVERGED      73 CALLS          74 TOTAL
                     EDM=8.18426e-08    STRATEGY= 1      ERROR MATRIX ACCURATE
  EXT PARAMETER                                   STEP         FIRST
  NO.   NAME      VALUE            ERROR          SIZE      DERIVATIVE
   1  Constant     4.57900e+04   8.82176e+01   7.06356e+00   4.05785e-06
   2  Mean         4.01899e+02   1.69177e-01   1.40074e-02   2.31593e-03
   3  Sigma        8.60719e+01   1.22068e-01   3.16271e-05  -1.51301e-01
```

FCN'e çok büyük! Eğri histograma yakışmadı

Eğri yakıştırmak 2

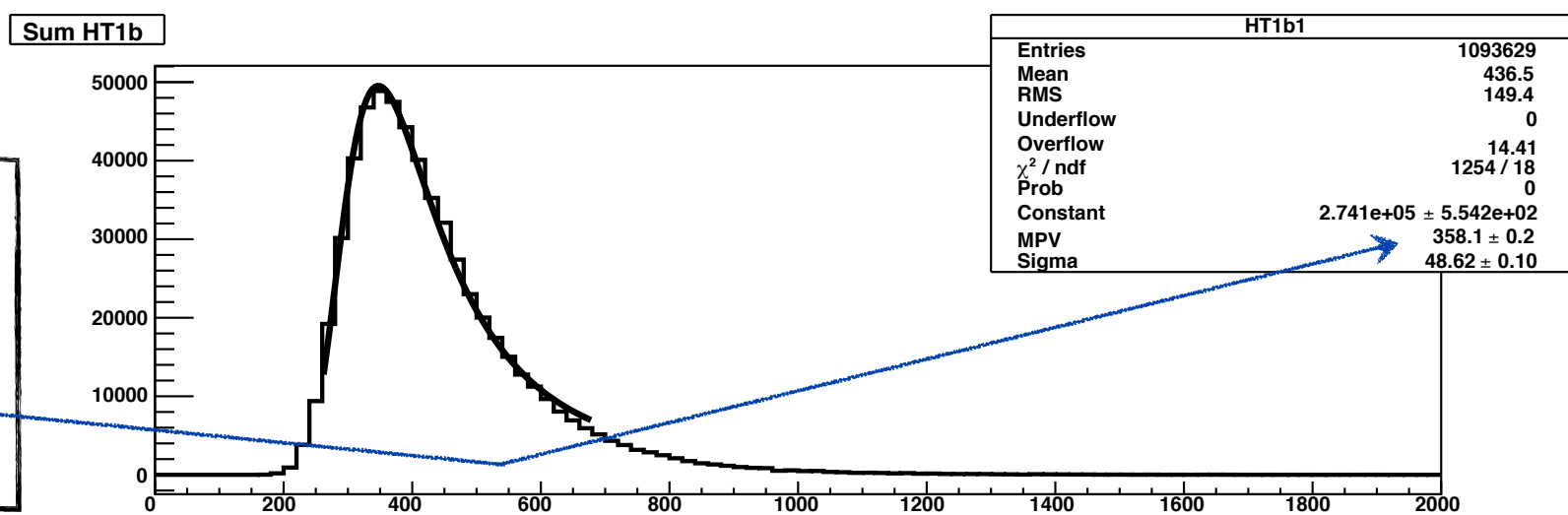
- Landau eğrisi ile denersek
  - pencereden seçin
  - daha iyi yakıştı...
  - landau 3 değişkenli
    - değerleri histogramdan tahmin edebiliriz:
      - Sabit : y_max/2 ≈ 25000
      - En Olası Değer ≈ 400
      - Genişlik : ≈ 50



- Betiğimize eklersek:

```
TF1 *egriA= new TF1("deneme","landau", 20, 1220);
     egriA->SetParameters(25000, 400, 50);
HT1b1->Fit("deneme","","",260, 680);
```

- çalıştırınca:



yakıştırma sonuçlarını ekrandan okuyalım:

Sabit: 27410

EOD : 358.1

Genişlik: 48.62

Eğri yakıştırmak 3

- Sonuçları ekrandan okumak yerine betikden alalım.
  - ✍TF1 *yakisan = HT1b1->GetFunction("deneme");
  - ✍Double_t eod = yakisan->GetParameter(1);
  - ✍cout << "EOD ="<<eod<<endl;
    - çalıştıralım

```
  EXT PARAMETER                                   STEP         FIRST
  NO.   NAME      VALUE            ERROR          SIZE      DERIVATIVE
   1  Constant     2.74114e+05   5.54151e+02  -3.50102e-03   1.86844e-08
   2  MPV          3.58069e+02   1.52083e-01  -7.06326e-04  -4.98699e-05
   3  Sigma        4.86204e+01   1.03874e-01  -6.40384e-07   4.33241e-02
EOD =358.069
Info in <TCanvas::Print>: pdf file dene1.pdf has been created
```

- Türkçeleştirebiliriz (Fit komutundan önce olmalı)
  - ✍egriA->SetParNames("Sabit","EOD","Genislik");
    - çalıştıralım

```
  NO.   NAME      VALUE            ERROR
   1  Sabit        2.74114e+05   5.54151e+02
   2  EOD          3.58069e+02   1.52083e-01
   3  Genislik     4.86204e+01   1.03874e-01
EOD =358.069
```

- Yakıştırma hatalarını da alalım
  - ✍Double_t eod_hata = yakisan->GetParError(1);
  - ✍cout << "EOD ="<<eod<< " +-" << eod_hata<<endl;
    - çalıştıralım

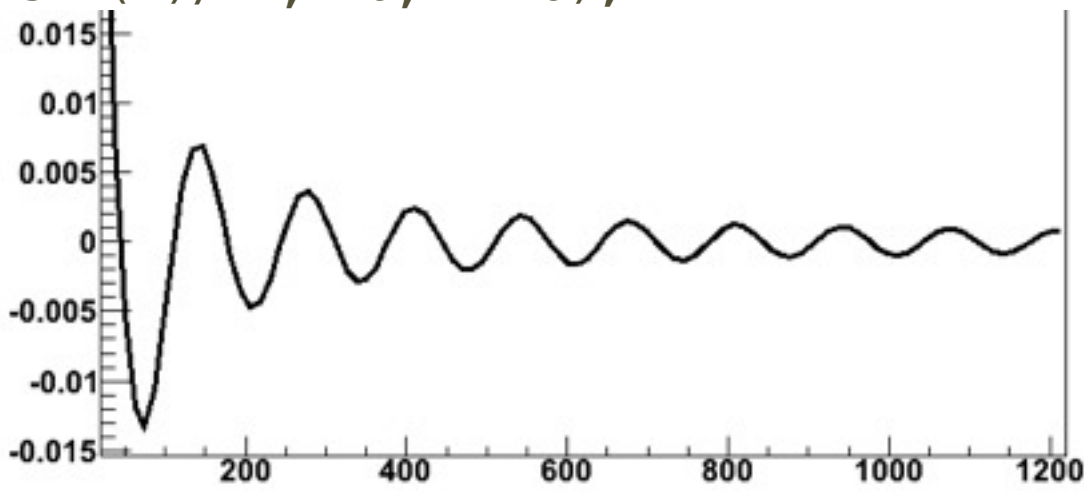
```
  NO.   NAME      VALUE            ERROR
   1  Sabit        2.74114e+05   5.54151e+02
   2  EOD          3.58069e+02   1.52083e-01
   3  Genislik     4.86204e+01   1.03874e-01
EOD =358.069 +-0.152083
```

Betiğin son durumu

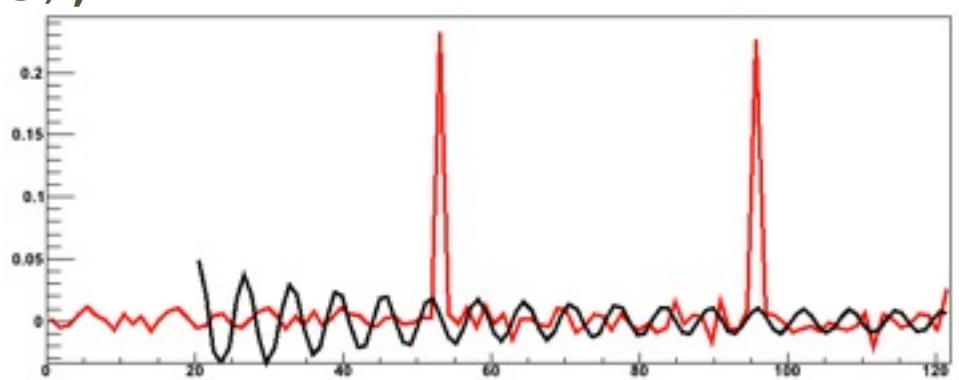
```
int dene1 (int xw=400, int yw=160)
{

TFile *_file0 = TFile::Open("105200.root");
FF_1->cd();
HTut1->SetLineColor(kBlue);
HTut1->SetLineStyle(2);

TCanvas *c1= new TCanvas("c1 ", "deneme", xw, yw);
c1->cd();

HTut1->Draw();
HT1b1->Draw("same");

TLatex lLumi;
lLumi.SetNDC(1);
TString TsLumi="#int L=9pb^{-1}";
lLumi.SetTextFont(42);
lLumi.SetTextSize(0.1);
lLumi.DrawLatex(0.6,0.4,TsLumi);


TF1 *egriA= new TF1("deneme","landau", 20, 1220);
     egriA->SetParameters(25000, 400, 50);
     egriA->SetParNames("Sabit","EOD","Genislik");
HT1b1->Fit("deneme","","",260, 680);
TF1 *yakisan = HT1b1->GetFunction("deneme");
Double_t eod = yakisan->GetParameter(1);
Double_t eod_hata = yakisan->GetParError(1);
cout << "EOD ="<<eod<< " +-" << eod_hata<<endl;

c1->SaveAs("dene1.pdf");
c1->SaveAs("dene1.jpg");
return 0;
}
```

1-boyutlu eğriler TF1

- Öntanımlı herhangi bir basit işlevi kullanabiliriz
  - ➡ `TF1 *egriB= new TF1("deneme","sin(x)/x", 20, 1220);`
  - ➡ `egriB->Draw();`



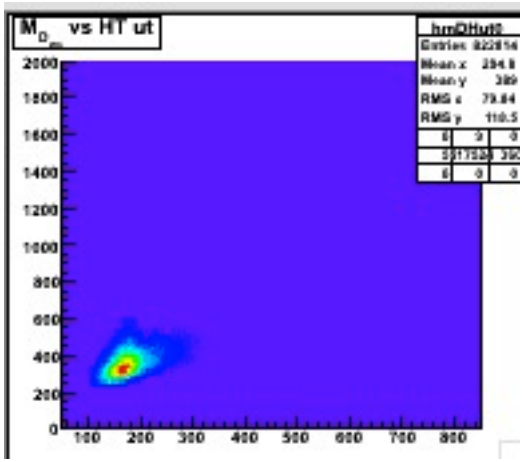
- Veya kendimiz tanımlayabiliriz - ancak betik içinde olmalı
  - ➡ `double abc(double *x, double *k){`
  - ➡ `return( k[0]*sin(x[0]) +k[1]/cos(x[0]**2)); }`
  - ➡ `TF1 *egriC= new TF1("deneme2",abc, 0, 122,2);`
  - ➡ `egriC->SetParameters(-0.002 , 0.003);`
  - ➡ `egriC->SetLineColor(2);`
  - ➡ `egriC->Draw();`
  - ➡ `egriB->Draw("same");`

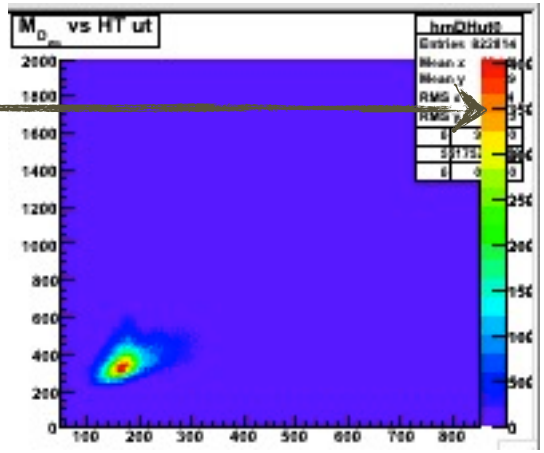


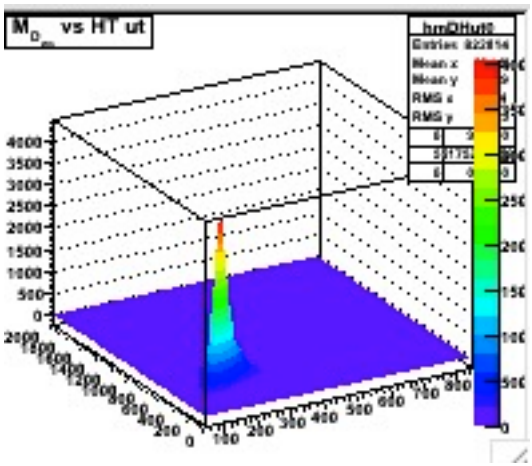
- ödev: TF2 nedir ?

TH2

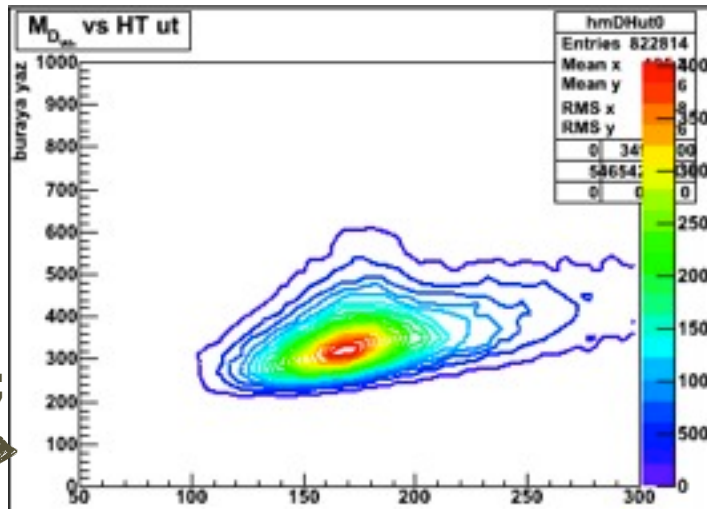
- kütükte gördüklerimize geri dönüyoruz.
  - ➡ `hmDHut0->Draw("col");`
    - renklerin anlamı:
  - ➡ `hmDHut0->Draw("colz");`
    - başka seçenekler de var
  - ➡ `hmDHut0->Draw("surf2z");`





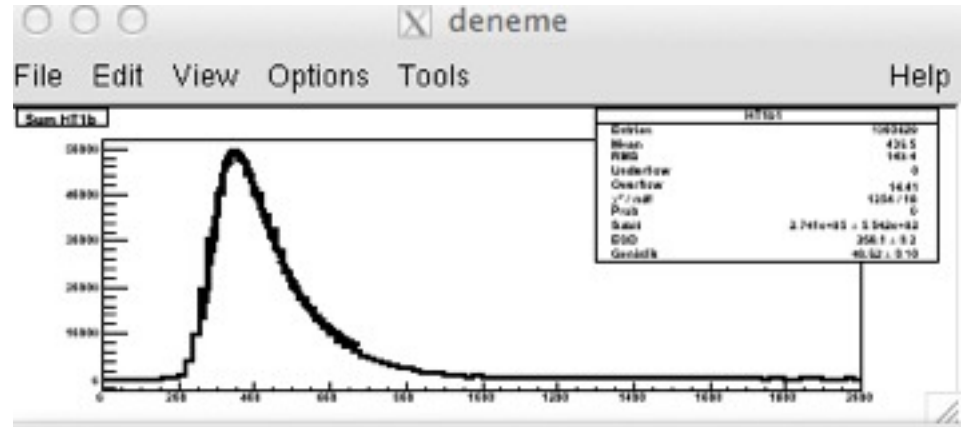


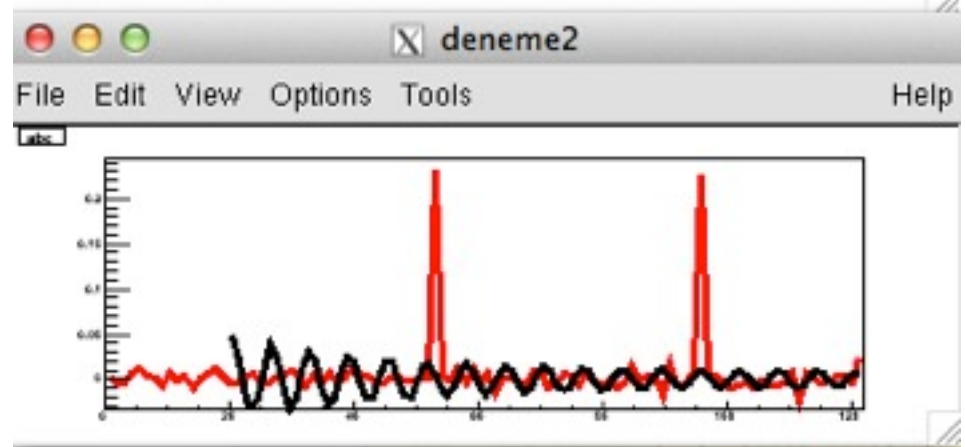
- Görünen tepeye yakından bakalım:
  - ➡ `hmDHut0->GetXaxis()->SetRange(0, 50);`
  - ➡ `hmDHut0->GetYaxis()->SetRange(0, 50);`
  - ➡ `hmDHut0->GetYaxis()->SetTitle("buraya yaz");`
  - ➡ `hmDHut0->Draw("cont1z");`

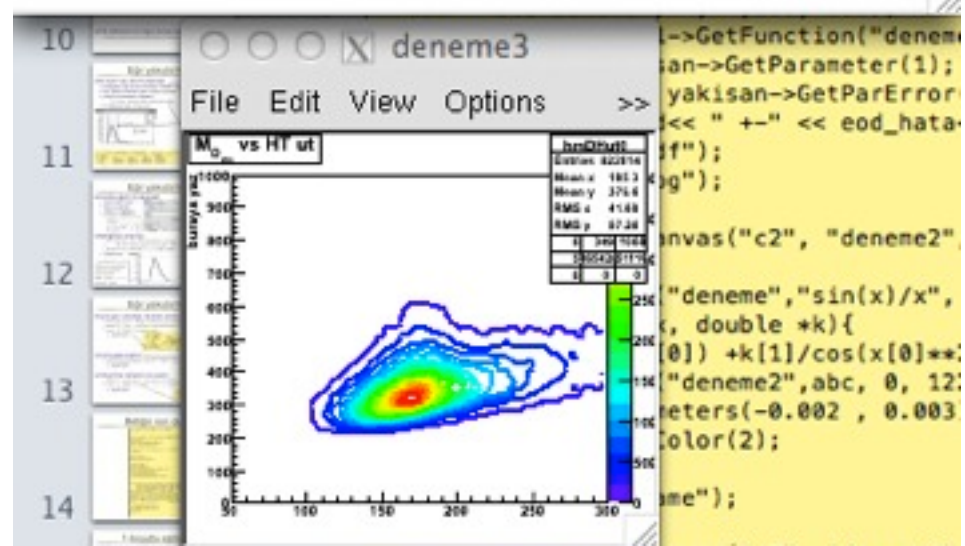


Betiğin son durumu

```
int dene1 (int xw=400, int yw=160) {
TFile *_file0 = TFile::Open("105200.root");
FF_1->cd();
HTut1->SetLineColor(kBlue);
HTut1->SetLineStyle(2);

TCanvas *c1= new TCanvas("c1 ", "deneme", xw, yw);
c1->cd();

HTut1->Draw();
HT1b1->Draw("same");

TLatex lLumi;
lLumi.SetNDC(1);
TString TsLumi="#int L=9pb^{-1}";
lLumi.SetTextFont(42);
lLumi.SetTextSize(0.1);
lLumi.DrawLatex(0.6,0.4,TsLumi);

TF1 *egriA= new TF1("deneme","landau", 20, 1220);
     egriA->SetParameters(25000, 400, 50);
     egriA->SetParNames("Sabit","EOD","Genislik");
HT1b1->Fit("deneme","","",260, 680);
TF1 *yakisan = HT1b1->GetFunction("deneme");
Double_t eod = yakisan->GetParameter(1);
Double_t eod_hata = yakisan->GetParError(1);
cout << "EOD ="<<eod<< " +-" << eod_hata<<endl;
c1->SaveAs("dene1.pdf");
c1->SaveAs("dene1.jpg");

TCanvas *c2= new TCanvas("c2", "deneme2", 400, 160);
c2->cd();
TF1 *egriB= new TF1("deneme","sin(x)/x", 20, 1220);
double abc(double *x, double *k){
 return( k[0]*sin(x[0]) +k[1]/cos(x[0]**2)); }
TF1 *egriC= new TF1("deneme2",abc, 0, 122,2);
     egriC->SetParameters(-0.002 , 0.003);
     egriC->SetLineColor(2);
     egriC->Draw();
     egriB->Draw("same");

TCanvas *c3= new TCanvas("c3", "deneme3", 200, 200);
c3->cd();
hmDHut0->GetXaxis()->SetRange(0, 50)
hmDHut0->GetYaxis()->SetRange(0, 50)
hmDHut0->GetYaxis()->SetTitle("buraya yaz");
hmDHut0->Draw("cont1z");

return 0; }
```







Olay döngüsü ve kayıt

- PF çalışmalarında her fizik olayı için kimi işleri tekrar tekrar yapmak gerekir. Örnek: her olaydaki jetleri say.

- histogram doldurmak
  - tanımla:

```
TH1F *hmultJ= new TH1F ("hmultJ","Jet Multiplicity",15,-0.5,14.5);
```

  - olay döngüsü kur:

```
for (Long64_t olay = 0; olay < olay_max; ++olay){
  hmultJ->Fill(njet, katsayi);
```

  - doldur

```
}
hmultJ->Draw("e");
```

  - çiz (e : hata işaretlerini göster)

```
TFile *bb = TFile::Open("dene2.root","recreate");
hmultJ->Write();
```

  - bir root kütüğüne yaz.

- bunları bir betiğe ekleyelim...

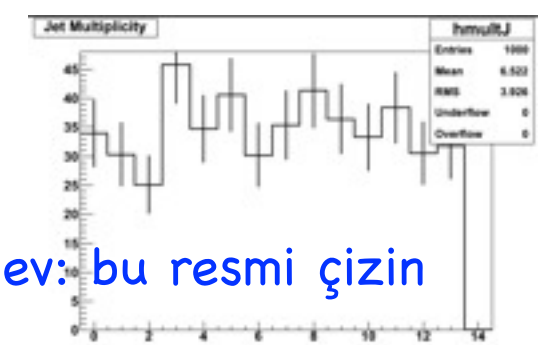
olay döngüsü betiği

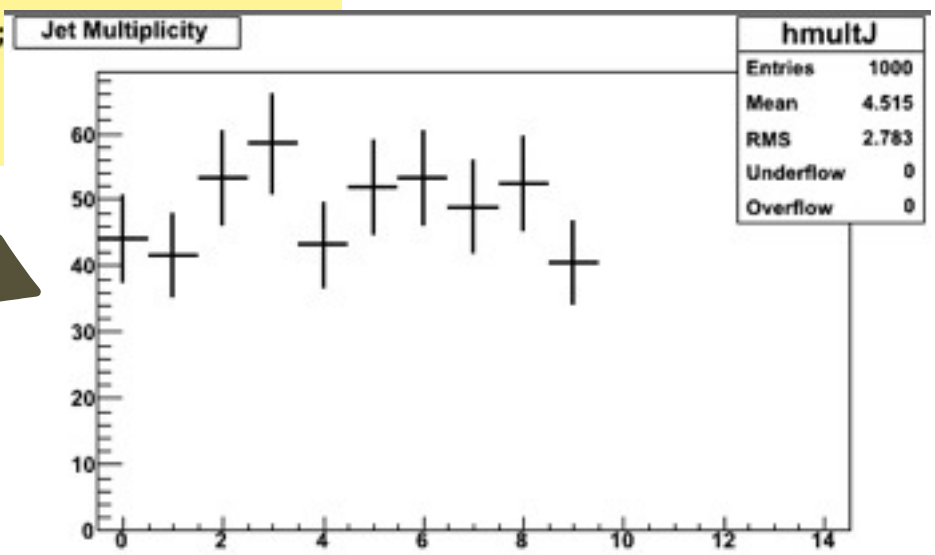
- Veri üretmeyi kolaylaştırmak için rastgele sayı üretelim
  - TRandom() kullanılabilir
  - Integer(10) 10'a kadar tamsayı üretmek için
  - Uniform(1) 0 ile 1 arasında tekdüze bir şekilde sayı üretir
- İlk defasında ekrana yazdıralım:

```
int dene2(int olay_max=10 ){
TRandom *uretici = new TRandom();
float katsayi, njet;
TH1F *hmultJ= new TH1F ("hmultJ","Jet Multiplicity",15,-0.5,14.5);
for (Long64_t olay = 0; olay < olay_max; ++olay){
  katsayi=uretici->Uniform(1.);
     njet=uretici->Integer(14);
  hmultJ->Fill(njet, katsayi);
//cout << "I: "<<njet << " k:"<<katsayi<<endl;
}
hmultJ->Draw("e");
TFile *bb = TFile::Open("dene2.root","recreate");
hmultJ->Write();
bb->Close();
return 0;}
```

```
Processing dene2.C...
I: 4 k:0.1543
I: 2 k:0.7906
I: 1 k:0.193708
I: 2 k:0.775951
I: 5 k:0.861995
I: 8 k:0.490301
I: 9 k:0.383607
I: 0 k:0.777661
I: 3 k:0.85014
I: 0 k:0.0288165
Info in <TCanvas::Make
```

➡ `root -x "dene2.C(1000)"`





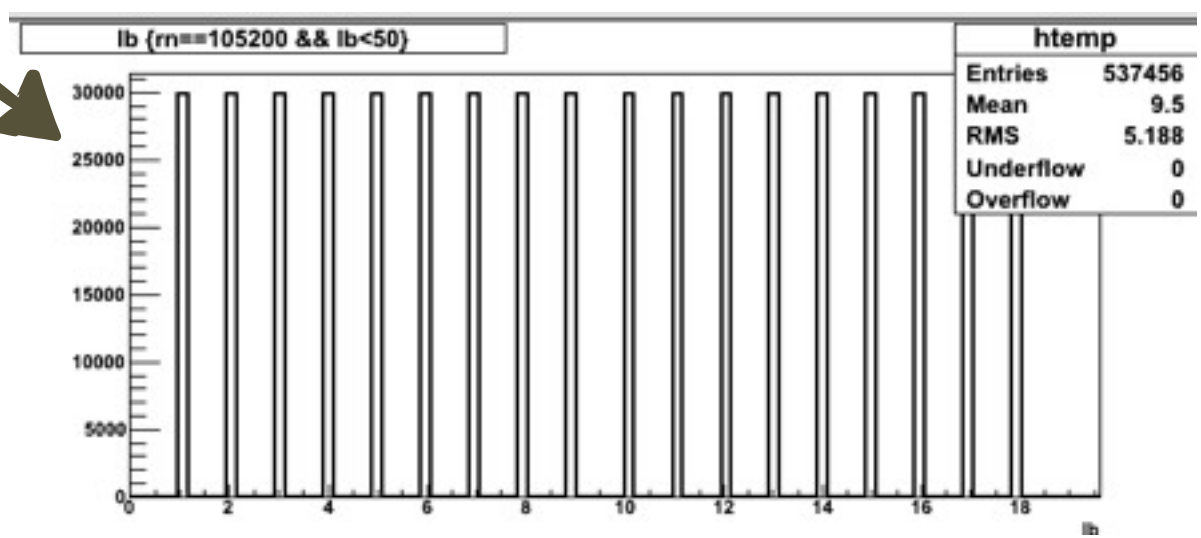
- ödev: bu resmi çizin

olay bilgisini saklamak: ntuple

- Olay başına bir çok bilgi içerir, ancak XL'den farklı
  - her satır bir olaysa, her satırda farklı sayıda sütun var.

- Örneğimize bakalım.
  - ➡ `rntuple->Scan()`
  - ➡ `rntuple->Scan("", "lb==18")`
    - lb=18 olanları sıralarsak...
  - ➡ `rntuple->Draw("lb","rn==105200")`
    - rn=105200 olan bütün lb leri çiz.
  - ➡ `rntuple->Draw("lb","rn==105200 && lb<50")`
    - rn=105200 ve 50den ufak olan bütün lb leri çiz.

```
root [3] rntuple->Scan()
************************************
*    Row   *        rn *        lb *
************************************
*        0 *    105200 *        18 *
*        1 *    105200 *        13 *
*        2 *    105200 *        10 *
*        3 *    105200 *        16 *
*        4 *    105200 *         3 *
*        5 *    105200 *       100 *
*        6 *    105200 *         4 *
*        7 *    105200 *        17 *
*        8 *    105200 *         1 *
*        9 *    105200 *         6 *
*       10 *    105200 *         8 *
*       11 *    105200 *         2 *
```



XL'in yaptığı basit işleri de...

- bir txt kütüğünden değer okuyup çizmek
  - mesela böyle bir kütükle başlayalım (t.txt)
  - bunu okuyup çizecek bir betik yazalım, işin özü:

✍ `TGraph *TA= new TGraph(N, xa, ya);`

1	2
2	4
3	9
4	16
5	25
6	36
7	49
8	64
9	81

```
int dene3 () {
float xa[99], ya[99];
float x,y;
int N=0;

ifstream ifs;
ifs.open("t.txt");

 while ( ifs.good() ) {
  ifs >> x >> y;
  cout <<"x:" << x<<" y:"<<y<<endl;
  xa[N]=x; ya[N]=y;
  N++;
 }
 ifs.close();

 gStyle->SetOptStat(0);
 TH2F *hpx = new TH2F("hpx"," :D ",40,0,10, 40,0,100);
 hpx->Draw();
 TGraph *TA= new TGraph(N, xa, ya);
 TA->Draw("*C");

TLegend *leg = new TLegend(0.5,0.15,0.89,0.24);
leg->AddEntry(TA,"Benim Egrim","l");
leg->Draw();

return 0; }
```