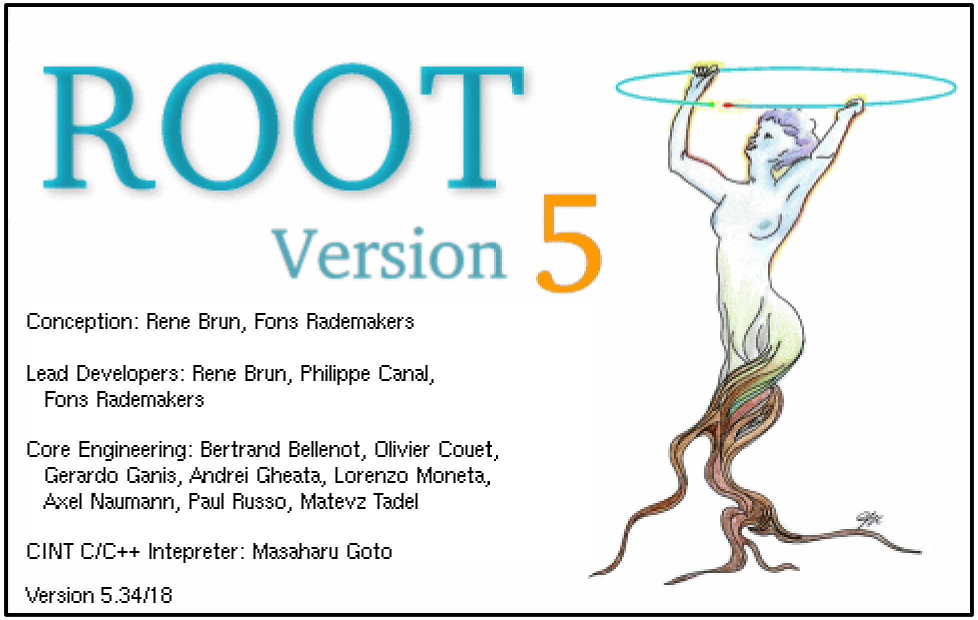
Introduction to ROOT



Fall 2014 - 2

Philip von Doetinchem
philipvd@hawaii.edu

Exercise

- Fill a histogram randomly (n=~10,000) with a Landau distribution with a most probable value at 20 and a “width” of 5 (use the ROOT website to find out about the Landau function)

- Fill the same histogram randomly (n=~5,000) with a Gaussian distribution centered at 5 with a “width” of 3

- Write a compiled script with a fit function that describes the total histogram nicely (it might be a good idea to fit both peaks individually first and use the fit parameters for a combined fit)

- Add titles to x- and y-axis

- Include a legend of the histogram with number of entries, mean, and RMS values

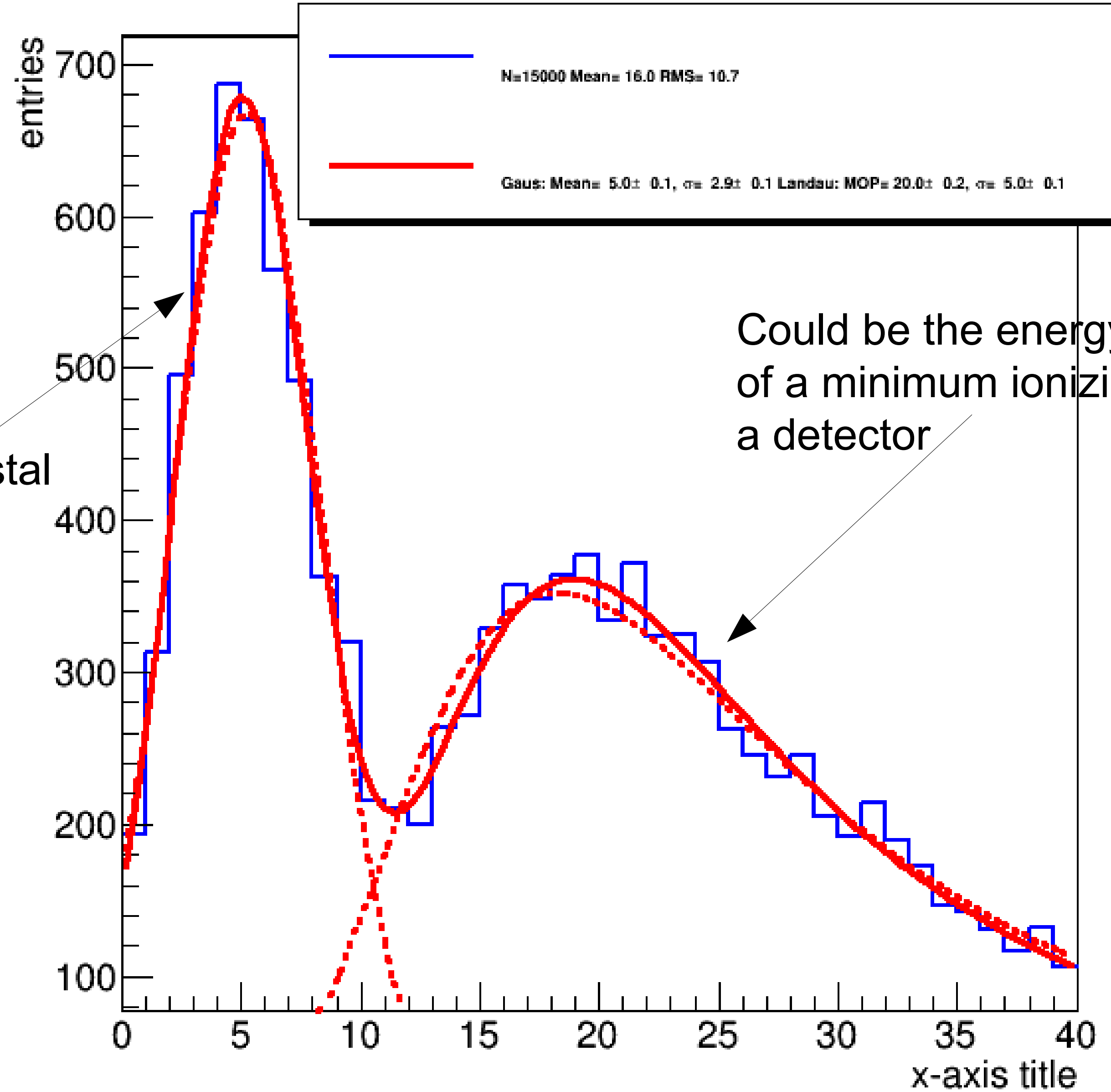
- Add text to the canvas with the fitted function parameters

- Draw everything on a square-size canvas (histogram in blue, fit in red)

- Save as png, eps, and root file

- Email me your result: philipvd@hawaii.edu

Exercise



Could be the electronics pedestal

Could be the energy deposition of a minimum ionizing particle in a detector

Solution exercise 1

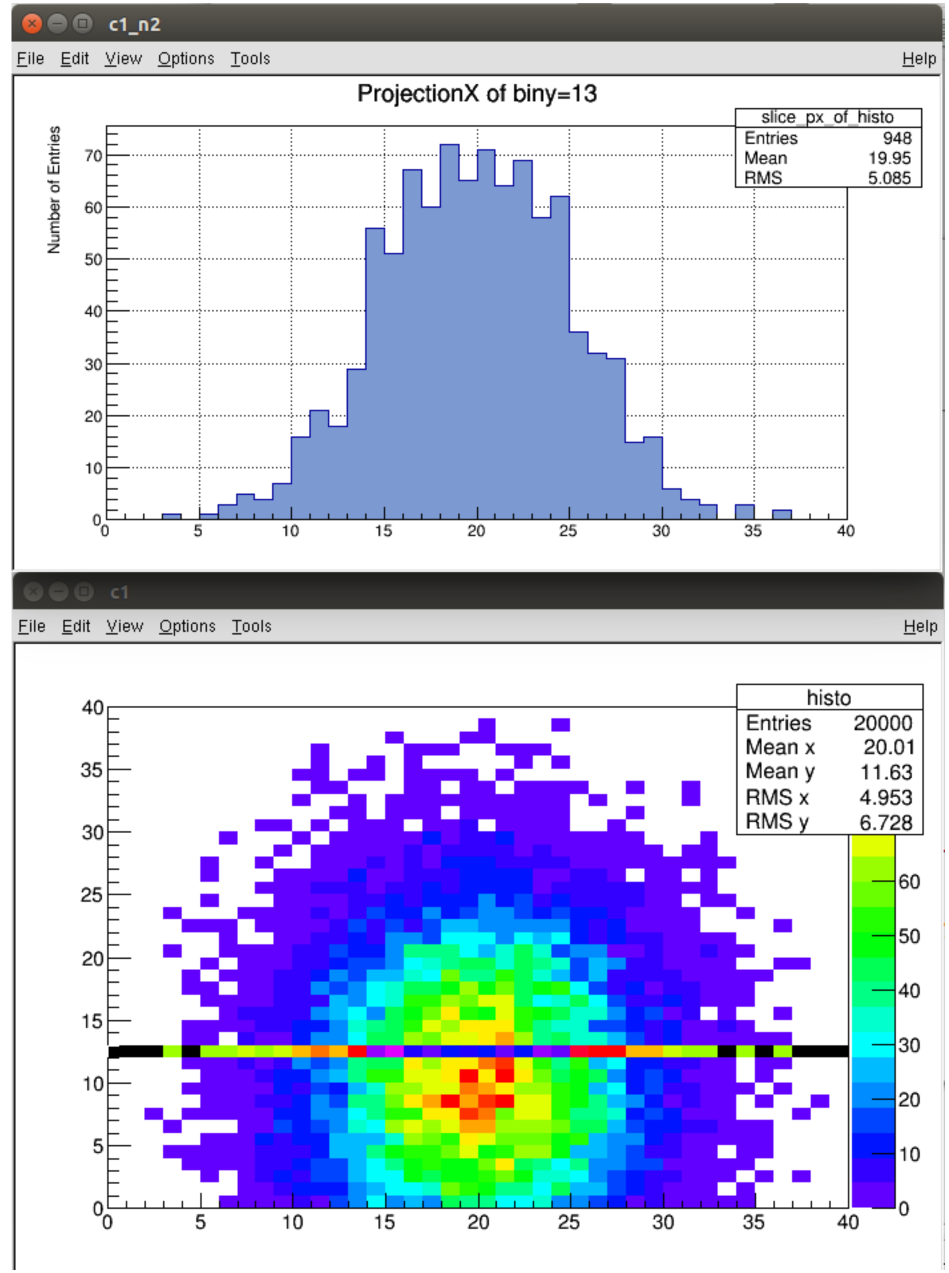
```
#include<iostream>

#include<TH1D.h>
#include<TF1.h>
#include<TCanvas.h>
#include<TRandom.h>
#include<TStyle.h>
#include<TLegend.h>
#include<TROOT.h>
#include<TPaveText.h>

void exercise1()
{
gROOT->Reset();
TStyle * plain = new TStyle("plain","plain");
plain->SetCanvasBorderMode(0);
plain->SetPadBorderMode(0);
plain->SetPadColor(0);
plain->SetCanvasColor(0);
plain->SetTitleColor(1);
plain->SetStatColor(0);
plain->SetTitleFillColor(0);

gROOT->SetStyle("plain");
gStyle->SetPalette(1);

//create empty histogram
TH1D * h = new TH1D("histo", "", 40,0,40);
//disable display of histogram statistics
h->SetStats(false);

//fill with Landau distribution
for(double i = 0; i < 10000; i++) h->Fill(gRandom->Landau(20,5));

//fill with Gaus distribution
for(double i = 0; i < 5000; i++) h->Fill(gRandom->Gaus(5,3));

//define fit functions
TF1 * FitFunc1 = new TF1("FitFunc1","[0]*TMath::Gaus(x,[1],[2])",0,40);
TF1 * FitFunc2 = new TF1("FitFunc2","[0]*TMath::Landau(x,[1],[2])",0,40);
TF1 * FitFuncCombined = new TF1("FitFunc2","[0]*TMath::Gaus(x,[1],[2])+[3]*TMath::Landau(x,[4],[5])",0,40);

//fit both peaks individually with reasonable initial parameters and fitting range
FitFunc1->SetParameters(1,3,4);
h->Fit(FitFunc1,"0","",0,10);

FitFunc2->SetParameters(1,17,7);
h->Fit(FitFunc2,"0","",10,40);
```

Solution exercise 1 - continued

```
//use fit parameters as initial parameters for combined fit
FitFuncCombined->SetParameters(FitFunc1->GetParameter(0), FitFunc1->GetParameter(1), FitFunc1->GetParameter(2), FitFunc2->GetParameter(0), FitFunc2->GetParameter(1), FitFunc2->GetParameter(2));
h->Fit(FitFuncCombined,"0","");

//display what we did
TCanvas * c = new TCanvas("c_ref","c_title", 200,10,600,600);
c->SetLeftMargin(0.15);
c->SetRightMargin(0.04);
c->SetTopMargin(0.04);

//Legend
TLegend* Leg = new TLegend(0.3,0.8,0.99,0.99);
Leg->SetFillColor(0);
Leg->SetTextFont(62);

h->SetLineWidth(2);
h->SetLineColor(kBlue);
h->GetXaxis()->SetTitle("x-axis title");
h->GetYaxis()->SetTitleOffset(1.4);
h->GetYaxis()->SetTitle("entries");
h->Draw();

char text[400];
sprintf(text,"N=%5.0f Mean=%5.1f RMS=%5.1f", h->GetEntries(), h->GetMean(), h->GetRMS());
Leg->AddEntry(h,text,"l");

FitFunc1->SetLineStyle(2);
FitFunc1->SetLineColor(kRed);
FitFunc1->Draw("same");

sprintf(text,"Gaus: Mean=%5.1f#pm%5.1f, #sigma=%5.1f#pm%5.1f Landau: MOP=%5.1f#pm%5.1f, #sigma=%5.1f#pm%5.1f", FitFuncCombined->GetParameter(1), FitFuncCombined->GetParError(1), FitFuncCombined->GetParameter(2), FitFuncCombined->GetParError(2), FitFuncCombined->GetParameter(4), FitFuncCombined->GetParError(4), FitFuncCombined->GetParameter(5), FitFuncCombined->GetParError(5));
Leg->AddEntry(FitFuncCombined,text,"l");

FitFunc2->SetLineStyle(2);
FitFunc2->SetLineColor(kRed);
FitFunc2->Draw("same");

FitFuncCombined->SetLineColor(kRed);
FitFuncCombined->Draw("same");

Leg->Draw();

//Save canvas
c->SaveAs("ex1.eps");
c->SaveAs("ex1.png");
c->SaveAs("ex1.root");
}
```

Projection

- Can be done interactively →
- extract projection from TH2 histogram



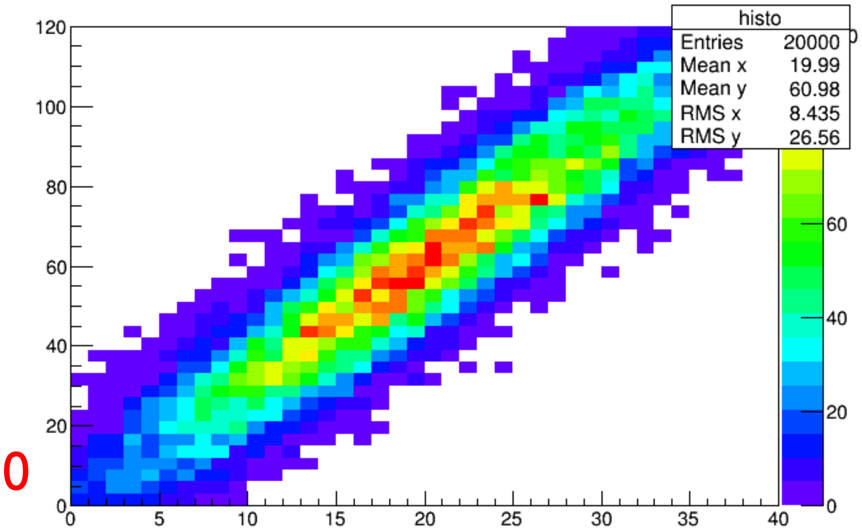

```
root [0] TH2D * h = new
TH2D("histo", "", 40,0,40,40,0,40);

root [1] for(double i = 0; i <
20000; i++) h->Fill(gRandom-
>Gaus(20,5), gRandom->Gaus(10,8));

root [2] TH1D * proj = h-
>ProjectionY(" ",21,21);

root [3] proj->Draw();
```

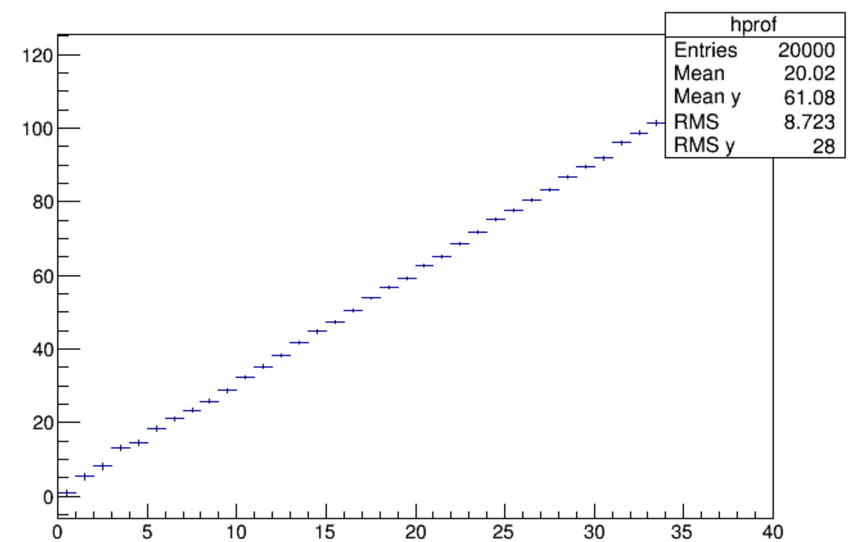
Works also for x

Choose bin range

Profile

```
root [0]  TH2D * h = new TH2D("histo", "", 40,0,40,40,0,120);
root [1] TProfile * hprof = new Tprofile("hprof","",40,0,40);

root [2] for(double i = 0; i < 20000; i++) { double x = gRandom->Gaus(20,10); double y = gRandom->Gaus(x*3+1,10); hprof->Fill(x,y); h->Fill(x,y);}

root [3] hprof->Draw();
root [4] h->Draw("colz")
```





The displayed error is by default the standard error on the mean (i.e. the standard deviation divided by the sqrt(n))

TMath and TRandom

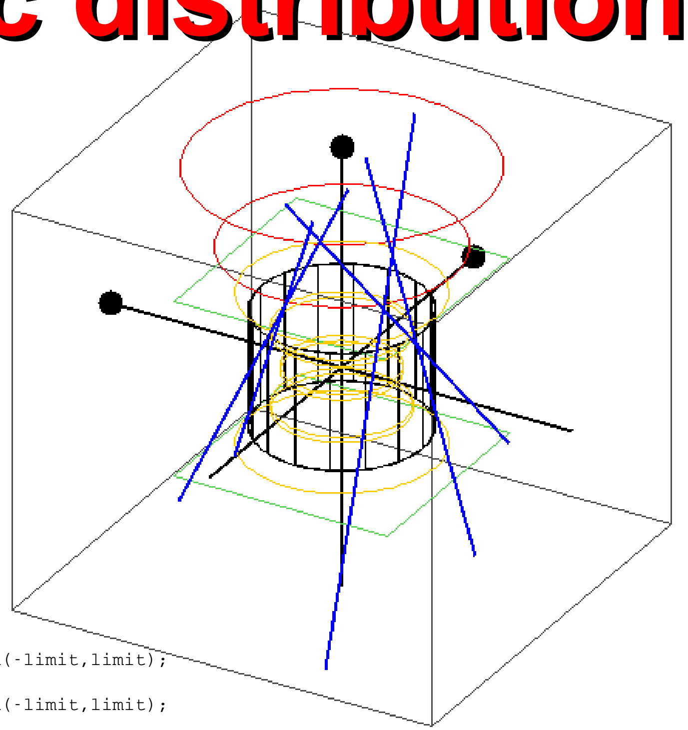
- TRandom: periodicity = 10^9 (fast)
  TRandom1: periodicity = 10^{171} (slow)
  TRandom2: periodicity = 10^{26} (fast)
  TRandom3: periodicity = 10^{6000} (fast)

```
TRandom3 * ran = new TRandom3();

ran->SetSeed(0); //0 = computer clock

ran->Uniform(1,10);
ran->Gaus(2,5);
```

TMatrix – inversion example

```
#include<TMatrixD.h>
#include<TDecompSVD.h>
#include<TVectorD.h>
#include<TRandom.h>

void matrix(int Layers = 4)
{
TMatrixD M(Layers, Layers);

for(int m = 0; m < Layers; m++)
for(int k = 0; k < Layers; k++) M[m]
[k] = gRandom->Gaus(5,10);

M.Print();

//Calculate inverse of the matrix via
singular value decomposition

TDecompSVD * SVD = new TDecompSVD(M);

TMatrixD U_T = SVD->GetU();
U_T.T();

TMatrixD V = SVD->GetV();

TVectorD S = SVD->GetSig();

//invert S

TMatrixD S_inv(Layers,Layers);
for(int i = 0; i < Layers; i++)
S_inv[i][i] = 1/S[i];

//multiply V S-1
TMatrixD VS_inv(Layers,Layers);

VS_inv.Mult(V,S_inv);

//calculate M-1
TMatrixD M_inv(Layers,Layers);

M_inv.Mult(VS_inv,U_T);

M_inv.Print();

//test
TMatrixD Test(Layers,Layers);
Test.Mult(M_inv,M);
Test.Print();
}
```

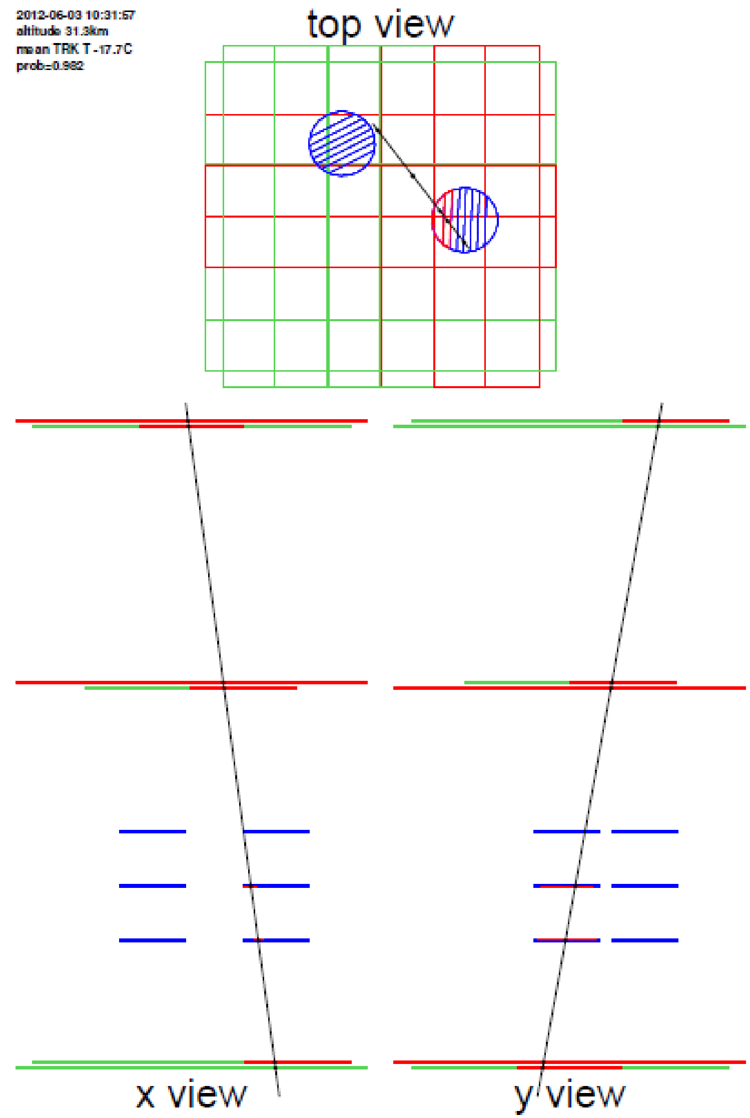
Cos^2 random isotropic distribution



```
TRandom3 * random = new TRandom3();
double phi = random->Uniform(0,2*TMath::Pi());

int plane = (rand()%6)+1;
double theta = acos(sqrt(random->Uniform(0,1)));
switch(plane)
      {
      case 1:
            {
            s1 = random->Uniform(-limit,limit);
            s2 = random->Uniform(-limit,limit);
            s3 = limit;

            n1 = sin(theta)*cos(phi);
            n2 = sin(theta)*sin(phi);
            n3 = cos(theta);
            }
      break;
      case 2:
            {
            s1 = random->Uniform(-limit,limit);
            s2 = random->Uniform(-limit,limit);
            s3 = -limit;

            n1 = sin(theta)*cos(phi);
            n2 = -sin(theta)*sin(phi);
            n3 = -cos(theta);
            }
      break;
      case 3:
            {
            s1 = random->Uniform(-limit,limit);
            s2 = limit;
            s3 = random->Uniform(-limit,limit);

            n1 = sin(theta)*cos(phi);
            n2 = -cos(theta);
            n3 = sin(theta)*sin(phi);
            }
      break;
        case 4:
            {
            s1 = random->Uniform(-limit,limit);
            s2 = -limit;
            s3 = random->Uniform(-limit,limit);

            n1 = -sin(theta)*cos(phi);
            n2 = cos(theta);
            n3 = sin(theta)*sin(phi);
            }
      break;
      case 5:
            {
            s1 = limit;
            s2 = random->Uniform(-limit,limit);
            s3 = random->Uniform(-limit,limit);

            n1 = cos(theta);
            n2 = sin(theta)*cos(phi);
            n3 = sin(theta)*sin(phi);
            }
      break;
      case 6:
            {
            s1 = -limit;
            s2 = random->Uniform(-limit,limit);
            s3 = random->Uniform(-limit,limit);

            n1 = -cos(theta);
            n2 = -sin(theta)*cos(phi);
            n3 = sin(theta)*sin(phi);
            }
      break;
      }
```

3D straight line fit

- Charged particle crosses several detectors as a straight line
- Imagine ~box-shaped detectors for charged particles arbitrarily rotated in x-y direction:



$$\vec{x} = \mathcal{R} \cdot \vec{u} \quad \text{with} \quad \mathcal{R} = \begin{pmatrix} \cos\alpha_{0,1} & -\sin\alpha_{0,1} & 0 \\ \sin\alpha_{0,1} & \cos\alpha_{0,1} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

- Covariance matrix of errors also has to be transformed
  → diagonal matrix is not any longer diagonal

$$\mathcal{U} = \mathcal{A} \cdot \mathcal{V} \cdot \mathcal{A}^T \quad \text{with} \quad \mathcal{V} = \begin{pmatrix} \sigma_u^2 & 0 & 0 \\ 0 & \sigma_v^2 & 0 \\ 0 & 0 & \sigma_w^2 \end{pmatrix} \quad \text{and} \quad \mathcal{A} = \begin{pmatrix} \frac{\partial x}{\partial u} & \frac{\partial x}{\partial v} & \frac{\partial x}{\partial w} \\ \frac{\partial y}{\partial u} & \frac{\partial y}{\partial v} & \frac{\partial y}{\partial w} \\ \frac{\partial z}{\partial u} & \frac{\partial z}{\partial v} & \frac{\partial z}{\partial w} \end{pmatrix}$$

- Try to minimize the distance of the measured hits to the straight line fit, respecting the errors

$$\chi^2 = \sum_i \chi_i^2 = \sum_i \vec{\Delta}_i^T \mathcal{U}^{-1} \vec{\Delta}_i$$

Chi2 minimization with Minuit

Global definition of Minuit parameters

```
vector<vector<double> > MinuitHit, MinuitError;
vector<double> MinuitAngle;
```

Chi2 calculation

```
double CalcChi2(double mx, double ax, double my, double ay)
{
double chi2 = 0;
for(unsigned int i = 0; i < MinuitHit.size(); i++)
      {
      double x = MinuitHit.at(i).at(0); double y = MinuitHit.at(i).at(1);
      double z = MinuitHit.at(i).at(2);
      double alpha = MinuitAngle.at(i);
      double sigmau = MinuitError.at(i).at(0); double sigmav =
      MinuitError.at(i).at(1); double sigmaw = MinuitError.at(i).at(2);

      double deltaxm = (mx*z+ax)-x; double deltaym = (my*z+ay)-y; double
      deltazm = 0;

      Chi2 += ...
      }
return chi2;
}
```

$$\chi^2 = \sum_i \chi_i^2 = \sum_i \vec{\Delta}_i^T \mathcal{U}^{-1} \vec{\Delta}_i$$

Chi2 minimization with Minuit

Fitting

```
MinuitHit = HitPos;
MinuitError = HitError; ← errors in detector coordinates
MinuitAngle = HitAngle;

TMinuit Minuit(4); ← initialize 4 track parameters
Minuit.SetFCN(TrackFitChi2); ← tell Minuit what to minimize
```

Fit and printout style

```
double Argument[10]; int Flag = 0;
Argument[0] = -1;
Minuit.mnexcm("SET PRint",Argument,1,Flag);
Argument[0] = 0;
Minuit.mnexcm("SET NOWarnings",Argument,1,Flag);
Argument[0] = 1;
Minuit.mnexcm("SET ERR", Argument,1,Flag);
```

Set starting values and step sizes for parameters

```
double mx = 1000; double ax = 0;
double my = 1000; double ay = 0;
Minuit.mnparm(0, "mx", mx, 0.01, 0,0,Flag);
Minuit.mnparm(1, "ax", ax, 0.01, 0,0,Flag);
Minuit.mnparm(2, "my", my, 0.01, 0,0,Flag);
Minuit.mnparm(3, "ay", ay, 0.01, 0,0,Flag);

Minuit.SetMaxIterations(500);
Minuit.Migrad();  ← actual Minuit fit
```

Access fit parameters

```
double emx, eax, emy, eay;
Minuit.GetParameter(0, mx, emx);
Minuit.GetParameter(1, ax, eax);
Minuit.GetParameter(2, my, emy);
Minuit.GetParameter(3, ay, eay);
```

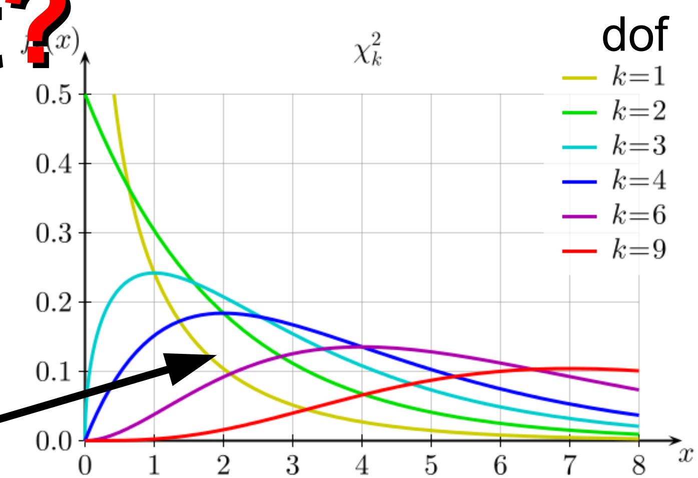
How good is your fit?

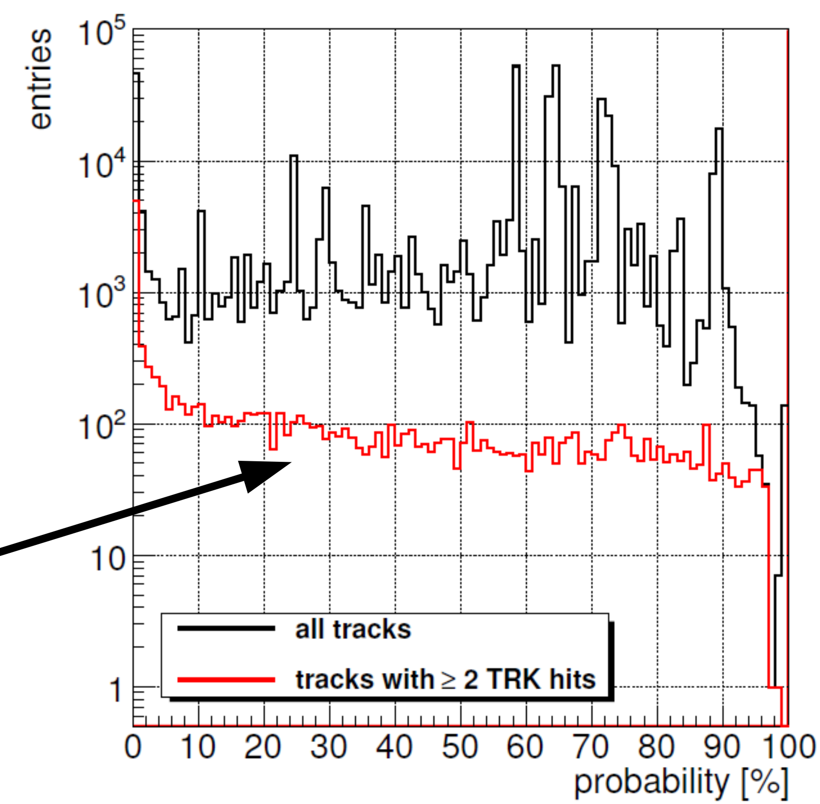
- Check chi2 → many people claim that chi2 per degree of freedom has to be 1 for a good fit
  - Its just rule of thumb
  - Chi2 follow a known statistical distribution, different for each number of degrees of freedom
  - What to do when you have a varying number of fit values?

- Use the cumulative chi2 distribution as a function of chi2 and dof → should be uniform:

$$p = \int_{\chi^2}^{\infty} f(t,n)\mathrm{d}t$$

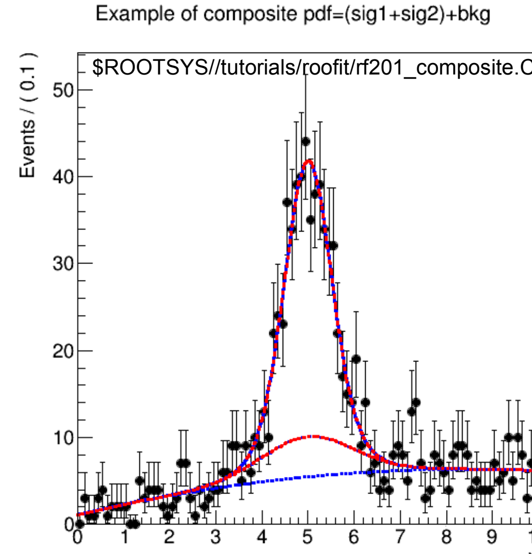
`TMath::Prob(chi2,dof)`



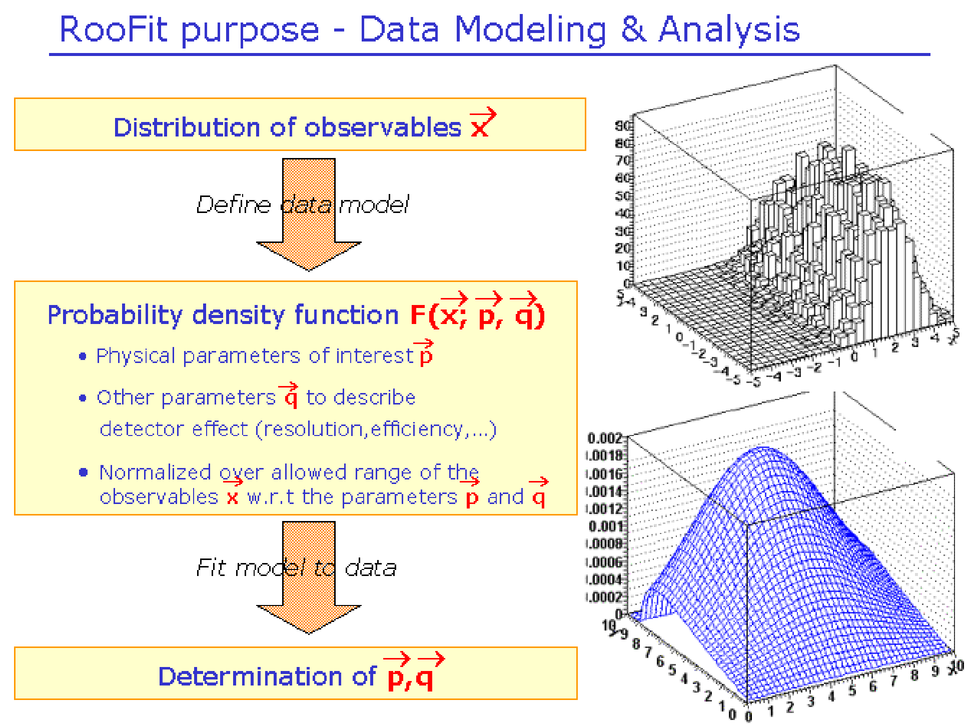


ROOFIT

./configure --enable-roofit



- The RooFit library provides a toolkit for modeling the expected distribution of events in a physics analysis
- core functionality of RooFit is to enable the modeling of ‘event data’ distributions, where each event is a discrete occurrence in time, and has one or more measured observables associated with it. Experiments of this nature result in datasets obeying Poisson (or binomial) statistics.



- Standard ROOT function framework clearly insufficient to handle such complicated functions
- Normalization of p.d.f. not always trivial to calculate
  → numeric integration techniques
- Simultaneous fit to control samples to account for detector performance

http://roofit.sourceforge.net/quicktour/first.html

Read ASCII file

- read ascii file with two columns

```
#include<fstream>
#include<iostream>
#include<TH2D.h>

void ascii(char *filename)
{
ifstream in(filename);

if(!in.good()) cout<<" "<<filename<<" does not
exist!"<<endl;
else
    {
    TH2D * h_frac = new TH2D("frac", "", 30,-50,50, 50,0,3);

    double x, y;
    while(1)
        {
        in>>x>>y;

        if(in.eof()) break;
        else
            {
            cout<<x<<" "<<y<<endl;
            h_frac->Fill(x,y);
            }
        }
    }
in.close();
}
```

Writing to file

```
> root -l
root [0] TFile * f = new
TFile("test12.root","recreate")
root [1] TH1D * h = new TH1D("histo", "",
100,0,1);
root [2] TF1 * f1 = new TF1("f1","Gaus(x,
[0],[1])",-1000,1000);
root [3] f1->SetParameters(0.5,0.1);
root [4] h->FillRandom("f1", 5000);
root [5] h->Write()
root [10] f->Close();
```

After opening a root file every object inherited from Tobject can be written to the file with a simple write statement.

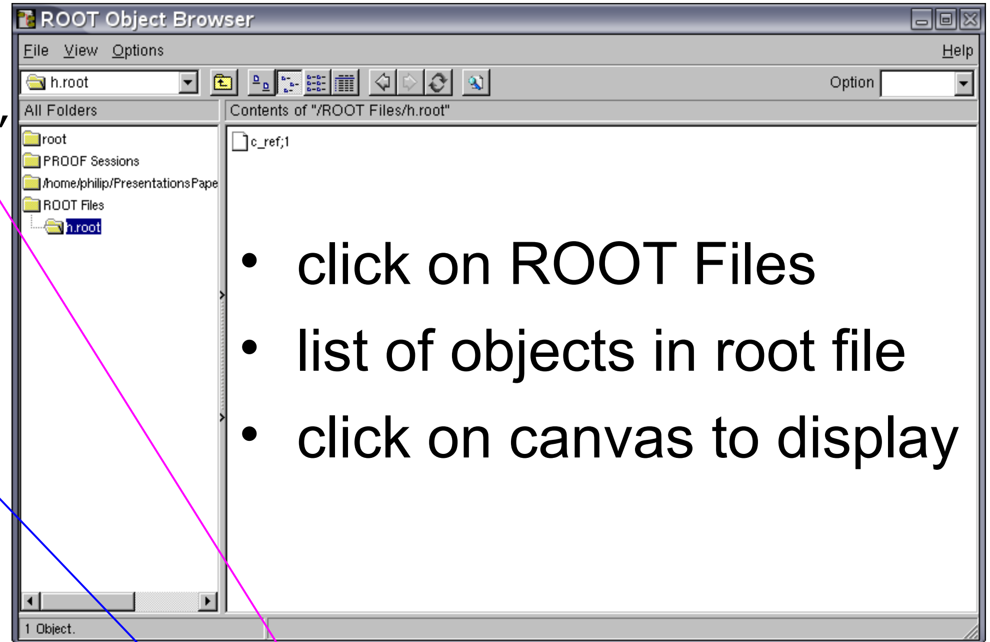
Open saved TCanvas/TBrowser

- save TCanvas with TH1D

```
TCanvas * c = new TCanvas("c_ref",
"c_title", 200,10,600,600);

TH1D * h = new TH1D("histo", "",
100,0,1);

TF1 * f1 = new TF1("f1","Gaus(x,
[0],[1])",-1000,1000);
f1->SetParameters(0.5,0.1);

h->FillRandom("f1", 5000);

h->Draw();

c->SaveAs("h.eps");
c->SaveAs("h.root");
```

- open the root file interactively

```
>root -l h.root
[0]
Attaching file h.root as _file0...
[1]new TBrowser
```



- click on ROOT Files
- list of objects in root file
- click on canvas to display

- open root file from script

```
TFile * file = new TFile("h.root");

TCanvas * c1 = (TCanvas*)file
->Get("c_ref");

TH1D * h1 = (TH1D*)c1
->GetListOfPrimitives()
->FindObject("histo");

file->Close();
```

Save and load TTree

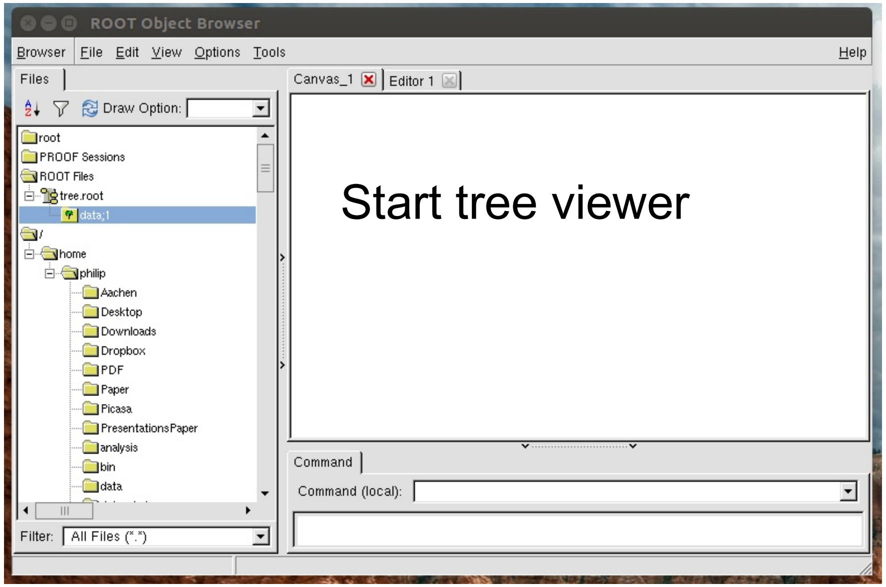
```
void save_tree()
{
TFile * f = new TFile("tree.root", "RECREATE");

TTree * t = new TTree("data","");

double x, y;
int event;

t->Branch("x", &x, "x/D");
t->Branch("y", &y, "y/D");
t->Branch("event", &event, "event/I");

TRandom3 * ran = new TRandom3();
ran->SetSeed(0);

for(int i = 0; i < 1000; i++)
     {
     x = ran->Gaus(0.5,3);
     y = ran->Exp(1);
     event = i;

     t->Fill();
     }

f->Write();
f->Close();
}
```

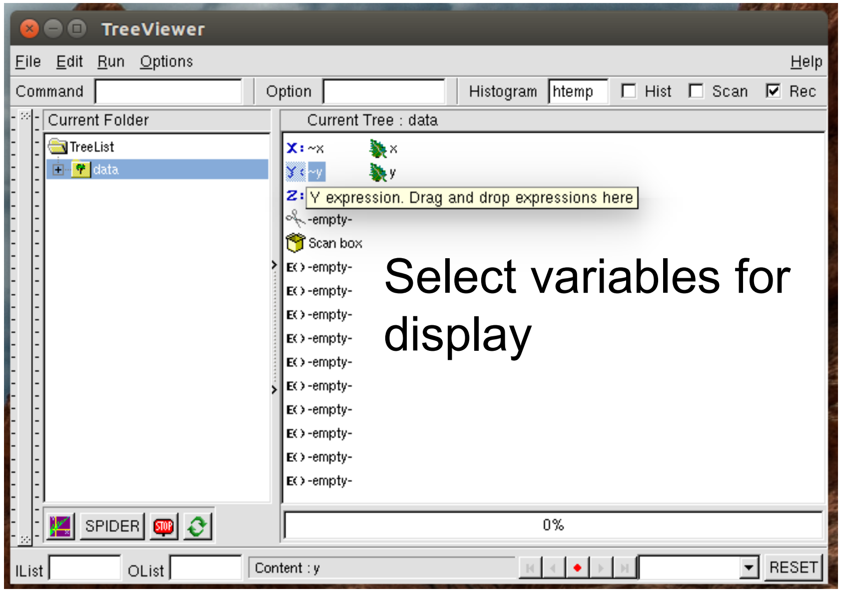
- save variables and all kind of inherited TObjects in root files

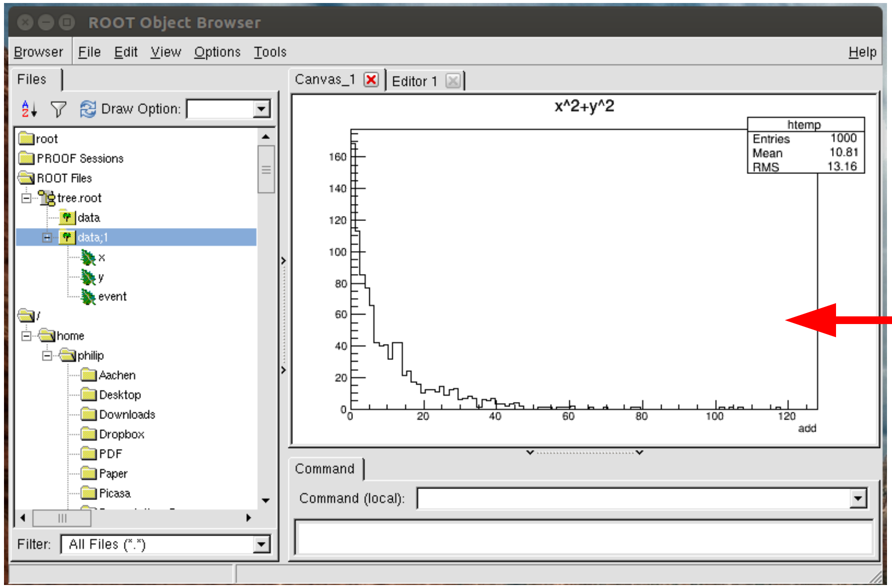
```
void open_tree()
{
TFile * f = new TFile("tree.root");

TTree * t = (TTree*)f->Get("data");

double x, y;
int event;

t->SetBranchAddress("x", &x);
t->SetBranchAddress("y", &y);
t->SetBranchAddress("event", &event);

for(Long64_t i = 0; i < t->GetEntriesFast(); i+
+)
     {
     t->GetEntry(i);
     cout<<event<<" "<<x<<" "<<y<<endl;
     }

f->Close();
}
```

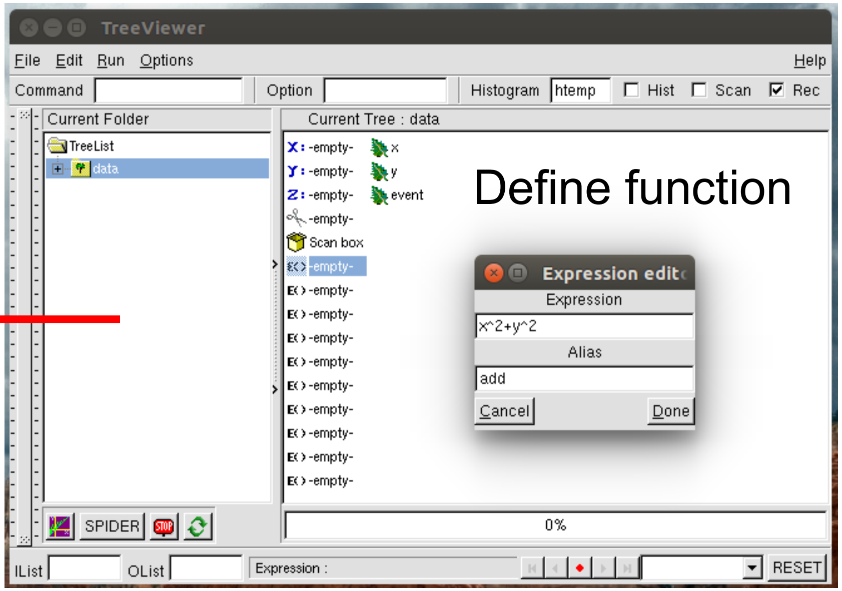
```
TH1D * h = new TH1D("h","",100,0,1);
h->Write("h");
```

Interactive tree viewer









TChain

```
TChain * TreeEvent = new TChain("Event");

int PrimaryParticlePdg = 0;
double PrimaryParticleEnergy = 0;
vector<int> * VertexTrackID = new vector<int>;

TreeEvent->SetBranchAddress("PrimaryParticlePdg",
&PrimaryParticlePdg);
TreeEvent-
>SetBranchAddress("PrimaryParticleEnergy",
&PrimaryParticleEnergy);

TreeEvent->SetBranchAddress("VertexTrackID",
&VertexTrackID);

TreeEvent->Add(“path-to-file/filename*123.root”);
```

Chains ROOT files of same type together. You can use wildcards with *.

Standalone programs

- C++ program:

```
#include<TApplication.h>
#include<TCanvas.h>
#include<TH1D.h>
#include<TF1.h>
#include<TStyle.h>
#include<TROOT.h>
#include<stdlib.h>

int main(int argc, char * argv[])
{
int n = atoi(argv[1]);

TApplication a("a", &argc, argv);

gROOT->Reset();

TStyle * plain = new TStyle("plain","plain");
plain->SetCanvasBorderMode(0);
plain->SetPadBorderMode(0);
plain->SetPadColor(0);
plain->SetCanvasColor(0);
plain->SetTitleColor(1);
plain->SetStatColor(0);
plain->SetTitleFillColor(0);

gROOT->SetStyle("plain");

TCanvas * c = new Tcanvas("c_ref","c_title",
 200,10,600,600);

TH1D * h = new TH1D("histo", "", 100,0,1);

TF1 * f1 = new TF1("f1","Gaus(x,[0],[1])",-1000,1000);
f1->SetParameters(0.5,0.1);

h->FillRandom("f1", n);

h->Draw();

a.Run();

return 0;
}
```

- Makefile:

```
ROOTLIBS     = $(shell root-config --libs)

all:         standalone

standalone:  standalone.o
             g++ -o standalone standalone.o $(ROOTLIBS)

.cpp.o:      $<.cpp
             g++ -I${ROOTSYS}/include -c $<

clean:
             @rm -f *.o
```

```
>make
>standalone 10000
```

Create new class

- header file: class.hpp

```
#ifndef ROOTCLASS
#define ROOTCLASS

#include "TObject.h"

class root_class : public TObject
{
public:

root_class();
~root_class();

double get_var1();
int get_var2();

void set_var1(double var1);
void set_var2(int var2);

private:

double its_var1;
int its_var2;

ClassDef(root_class,1)
};

#endif
```

- function definitions: class.cpp

```
#include "class.hpp"

root_class::root_class()
{
its_var1 = 5.3;
its_var2 = 3;
}

root_class::~root_class(){;}

double root_class::get_var1(){return its_var1;}

int root_class::get_var2(){return its_var2;}

void root_class::set_var1(double var1){its_var1 =
var1;}

void root_class::set_var2(int var2){its_var2 = var2;}
```

- link new class to root: linkdef.hpp

```
#ifdef __CINT__

#pragma link C++ class root_class+;

#endif
```

Save and read new class

- Makefile

```
ROOTLIBS    = $(shell root-config --libs)

all:        class

class: main_class.o class.o dict.o
            g++ -o class main_class.o class.o dict.o $
(ROOTLIBS)

dict.cpp: class.hpp class.cpp
            rootcint -f dict.cpp -c class.hpp linkdef.hpp

.cpp.o:$<.cpp
            g++ -I${ROOTSYS}/include -c $<

clean:
            @rm -f *.o dict.h dict.cpp
```

- excerpt of main file: main_class.cpp

```
root_class * c1 = new root_class();

TFile * f1 = new TFile("root_class.root", "RECREATE");
TTree * t1 = new TTree("class1","");
t1->Branch("c1", &c1);

for(int i = 0; i < 1000; i++)
      {
      c1->set_var1(sqrt(double(i)));
      c1->set_var2(i-3);
      t1->Fill();
      }

f1->Write();
f1->Close();

root_class * c2 = new root_class();
TChain * t2 = new TChain("class1");
t2->SetBranchAddress("c1",&c2);
t2->Add("root_class.root");

for(Long64_t i = 0; i < t2->GetEntriesFast(); i++)
      {
      t2->GetEntry(i);
      cout<<c2->get_var1()<<" "<<c2->get_var2()<<endl;
      }
```

TThread

```
vector<TThread*> t_beta_ecal_rigidity_deuteron;
```

← create vector of thread pointers

```
vector<TH2D*> h_beta_ecal_rigidity_deuteron;
for(int i = 0; i < 4; i++)
```

Number of threads

```
       {
       char text[400];
       sprintf(text,"beta_ecal_rigidity_deuteron_%d", i);
       h_beta_ecal_rigidity_deuteron.push_back(new TH2D(text, "",
       20,0.6,1,1000,0,1));

       sprintf(text,"thread_beta_ecal_rigidity_deuteron_%d", i);
       t_beta_ecal_rigidity_deuteron.push_back(new TThread(text,
       fill_beta_ecal_rigidity_deuteron, (void*)
       h_beta_ecal_rigidity_deuteron.at(i)));
```

← create threads, each thread fills a histogram

```
       t_beta_ecal_rigidity_deuteron.at(i)->Run();
```

← run threads

```
       }

for(unsigned int i = 0; i < t_beta_ecal_rigidity_deuteron.size(); i++)
t_beta_ecal_rigidity_deuteron.at(i)->Join();
```

← wait until all threads are done

```
for(unsigned int i = 0; i < h_beta_ecal_rigidity_deuteron.size(); i++)
h_beta_ecal_rigidity_deuteron_total-
>Add(h_beta_ecal_rigidity_deuteron.at(i));
```

← merge histograms

TThread

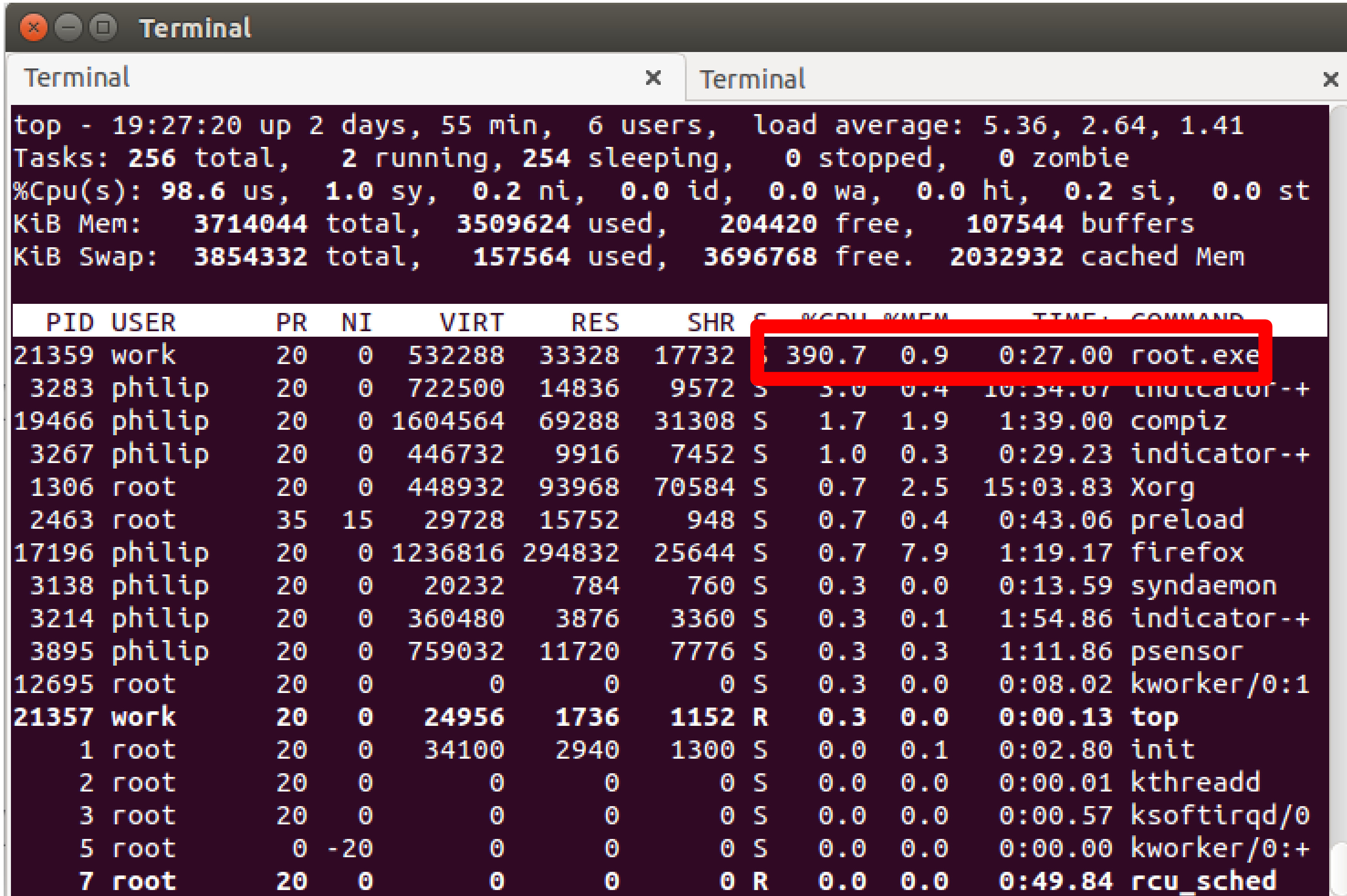
r of thread

, "",

i);

ext,

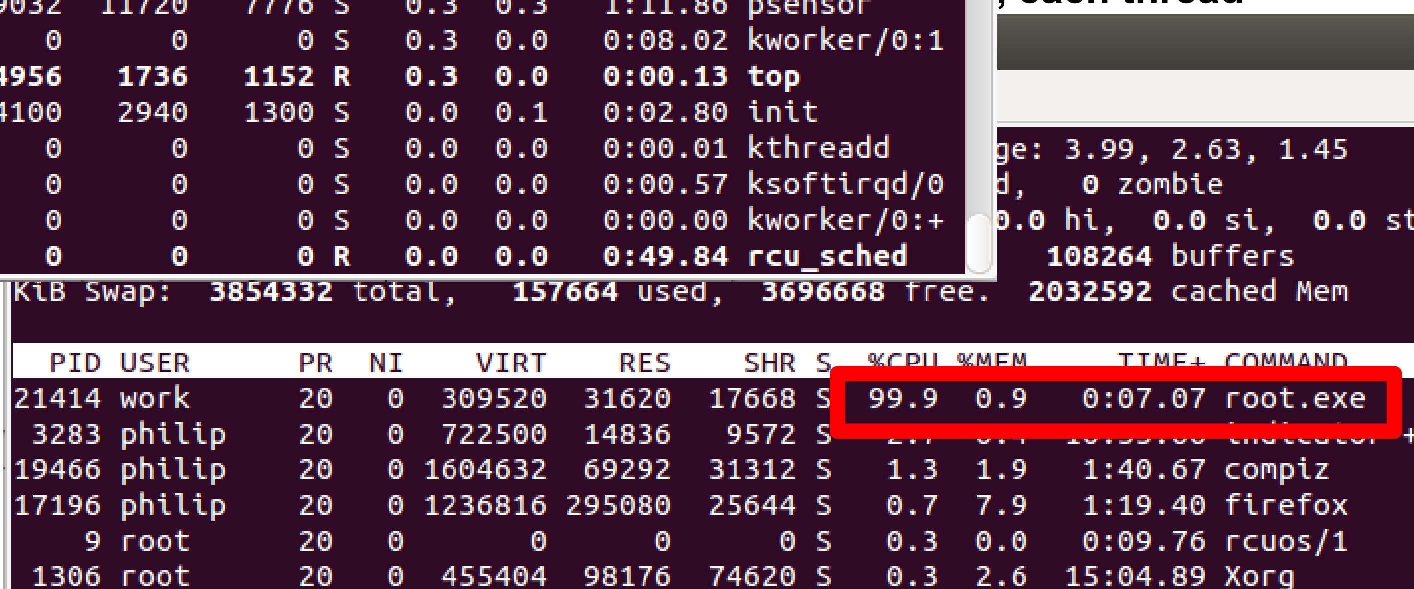
, each thread



```
Terminal
Terminal    Terminal
top - 19:27:20 up 2 days, 55 min,  6 users,  load average: 5.36, 2.64, 1.41
Tasks: 256 total,   2 running, 254 sleeping,   0 stopped,   0 zombie
%Cpu(s): 98.6 us,  1.0 sy,  0.2 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.2 si,  0.0 st
KiB Mem:   3714044 total,  3509624 used,   204420 free,   107544 buffers
KiB Swap:  3854332 total,   157564 used,  3696768 free.  2032932 cached Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
21359 work      20   0  532288  33328  17732 S 390.7  0.9   0:27.00 root.exe
 3283 philip    20   0  722500  14836   9572 S   3.0  0.4  10:34.07 indicator-+
19466 philip    20   0 1604564  69288  31308 S   1.7  1.9   1:39.00 compiz
 3267 philip    20   0  446732   9916   7452 S   1.0  0.3   0:29.23 indicator-+
 1306 root      20   0  448932  93968  70584 S   0.7  2.5  15:03.83 Xorg
 2463 root      35  15   29728  15752    948 S   0.7  0.4   0:43.06 preload
17196 philip    20   0 1236816 294832  25644 S   0.7  7.9   1:19.17 firefox
 3138 philip    20   0   20232    784    760 S   0.3  0.0   0:13.59 syndaemon
 3214 philip    20   0  360480   3876   3360 S   0.3  0.1   1:54.86 indicator-+
 3895 philip    20   0  759032  11720   7776 S   0.3  0.3   1:11.86 psensor
12695 root      20   0       0      0      0 S   0.3  0.0   0:08.02 kworker/0:1
21357 work      20   0   24956   1736   1152 R   0.3  0.0   0:00.13 top
    1 root      20   0   34100   2940   1300 S   0.0  0.1   0:02.80 init
    2 root      20   0       0      0      0 S   0.0  0.0   0:00.01 kthreadd
    3 root      20   0       0      0      0 S   0.0  0.0   0:00.57 ksoftirqd/0
    5 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/0:+
    7 root      20   0       0      0      0 R   0.0  0.0   0:49.84 rcu_sched
```

t_beta_ecal_rigidi

```
for(unsigned int i
h_beta_ecal_rigidi
>Add(h_beta_ecal_r
```



```
ge: 3.99, 2.63, 1.45
d,   0 zombie
0.0 hi,  0.0 si,  0.0 st
108264 buffers
KiB Swap:  3854332 total,   157664 used,  3696668 free.  2032592 cached Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
21414 work      20   0  309520  31620  17668 S  99.9  0.9   0:07.07 root.exe
 3283 philip    20   0  722500  14836   9572 S
19466 philip    20   0 1604632  69292  31312 S   1.3  1.9   1:40.67 compiz
17196 philip    20   0 1236816 295080  25644 S   0.7  7.9   1:19.40 firefox
    9 root      20   0       0      0      0 S   0.3  0.0   0:09.76 rcuos/1
 1306 root      20   0  455404  98176  74620 S   0.3  2.6  15:04.89 Xorg
```

PROOF

The Parallel ROOT Facility, PROOF, is an extension of ROOT enabling interactive analysis of large sets of ROOT files in parallel on clusters of computers or many-core machines.

- PROOF stands for Parallel ROOt Facility
- It allows parallel processing of large amount of data. The output results can be directly visualised (e.g. the output histogram can be drawn at the end of the proof session).
- PROOF is NOT a batch system.
- The data which you process with PROOF can reside on your computer, PROOF cluster disks or grid.
- The usage of PROOF is transparent: you should not rewrite your code you are running locally on your computer.
- No special installation of PROOF software is necessary to execute your code: PROOF is included in ROOT distribution.

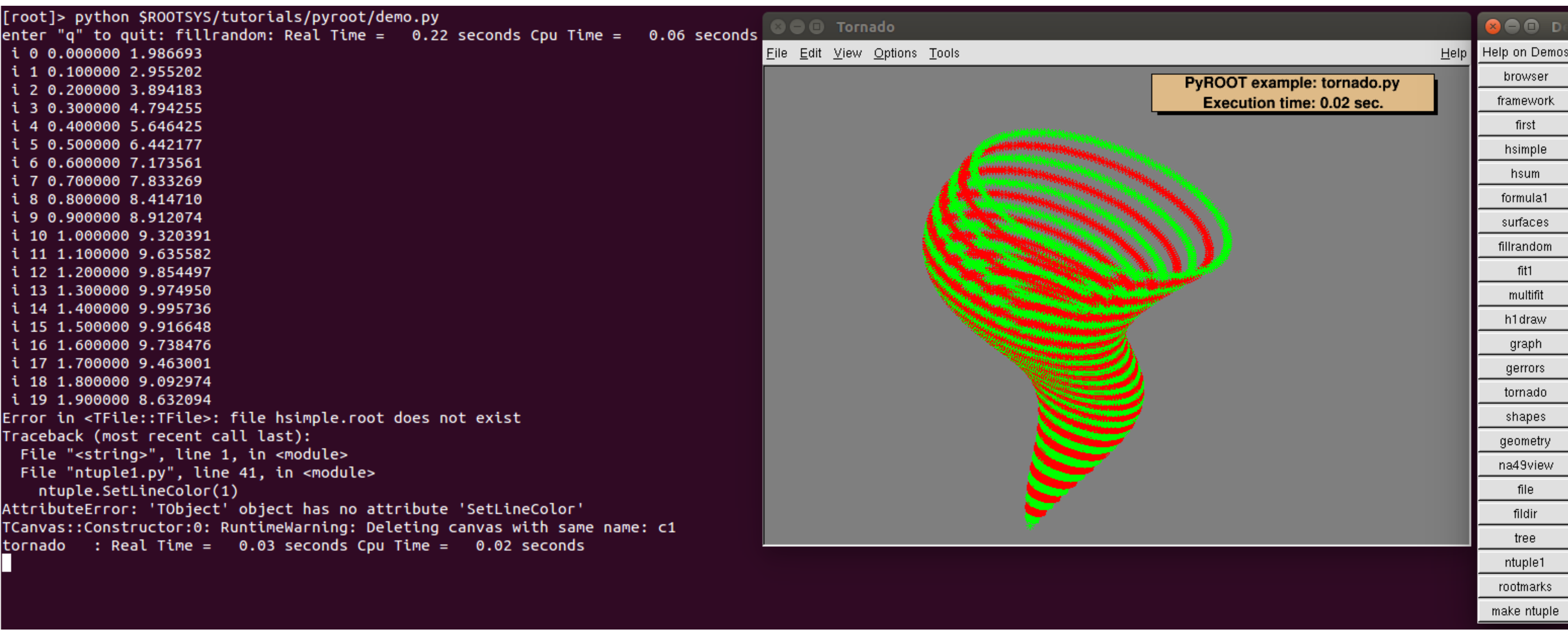
pyROOT

Add to .tcshrc (change accordingly for bash)

```
setenv PYTHONDIR /usr/include/python2.7
```
(← check which Python was used at compilation time)
```
setenv LD_LIBRARY_PATH $ROOTSYS/lib:$PYTHONDIR/lib:$LD_LIBRARY_PATH
setenv PYTHONPATH $ROOTSYS/lib
```

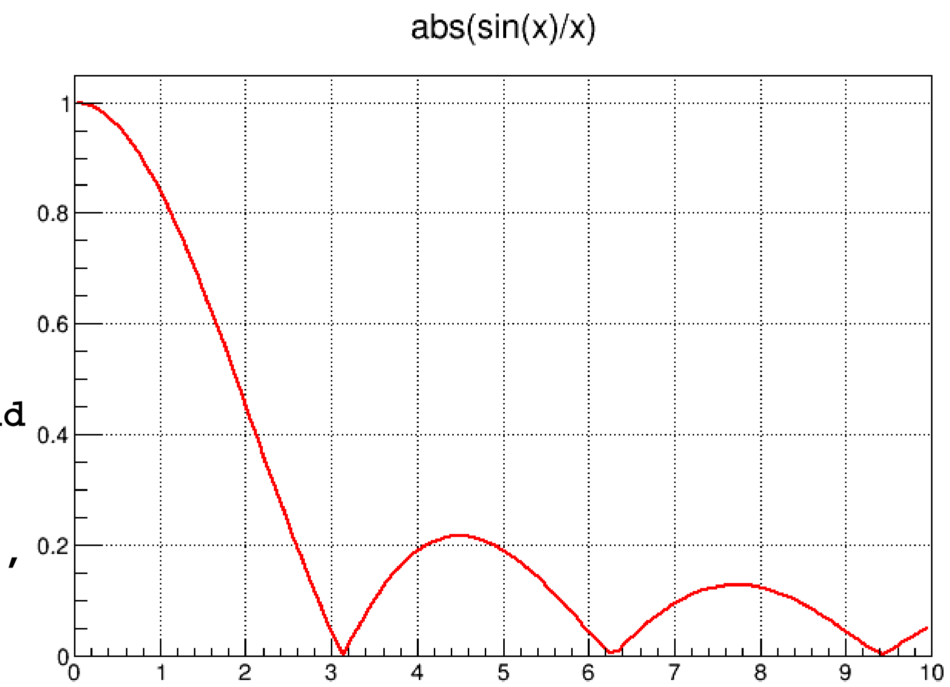
Example:

```
>python $ROOTSYS/tutorials/pyroot/demo.py
```



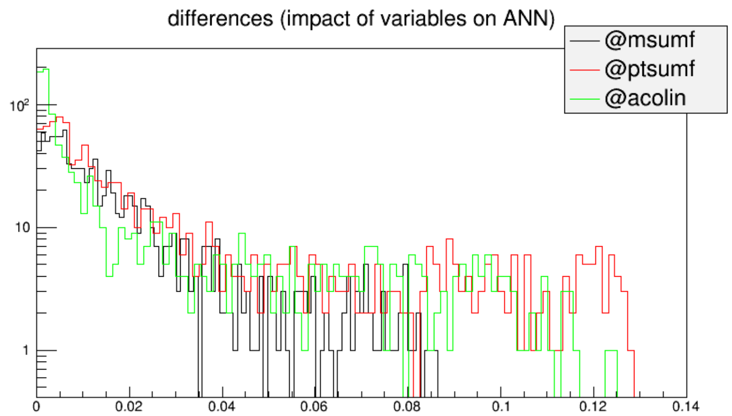
PyROOT

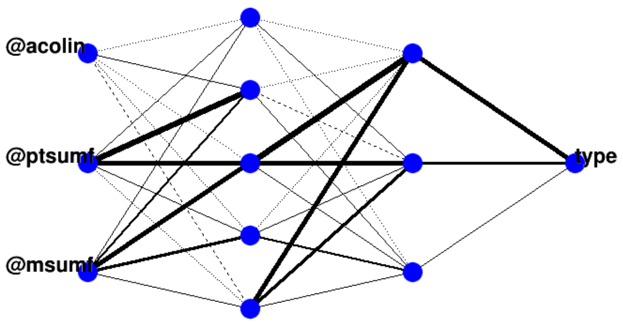
```
from ROOT import gROOT, TCanvas, TF1

gROOT.Reset()
c1 = TCanvas( 'c1', 'Example with
Formula', 200, 10, 700, 500 )

#
# Create a one dimensional function and
draw it
#
fun1 = TF1( 'fun1', 'abs(sin(x)/x)', 0,
10 )
c1.SetGridx()
c1.SetGridy()
fun1.Draw()
c1.Update()

## wait for input to keep the GUI (which
lives on a ROOT event dispatcher) alive
if __name__ == '__main__':
   rep = ''
   while not rep in [ 'q', 'Q' ]:
      rep = raw_input( 'enter "q" to
quit: ' )
      if 1 < len(rep):
         rep = rep[0]
```

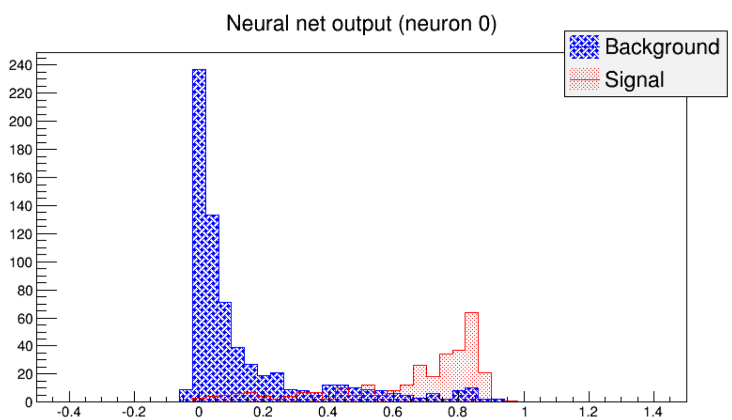


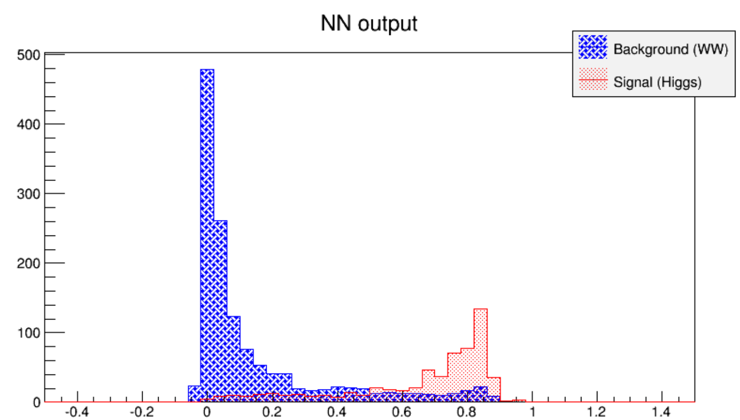
Neural network

$ROOTSYS/tutorials/mlp/mlpHiggs.C









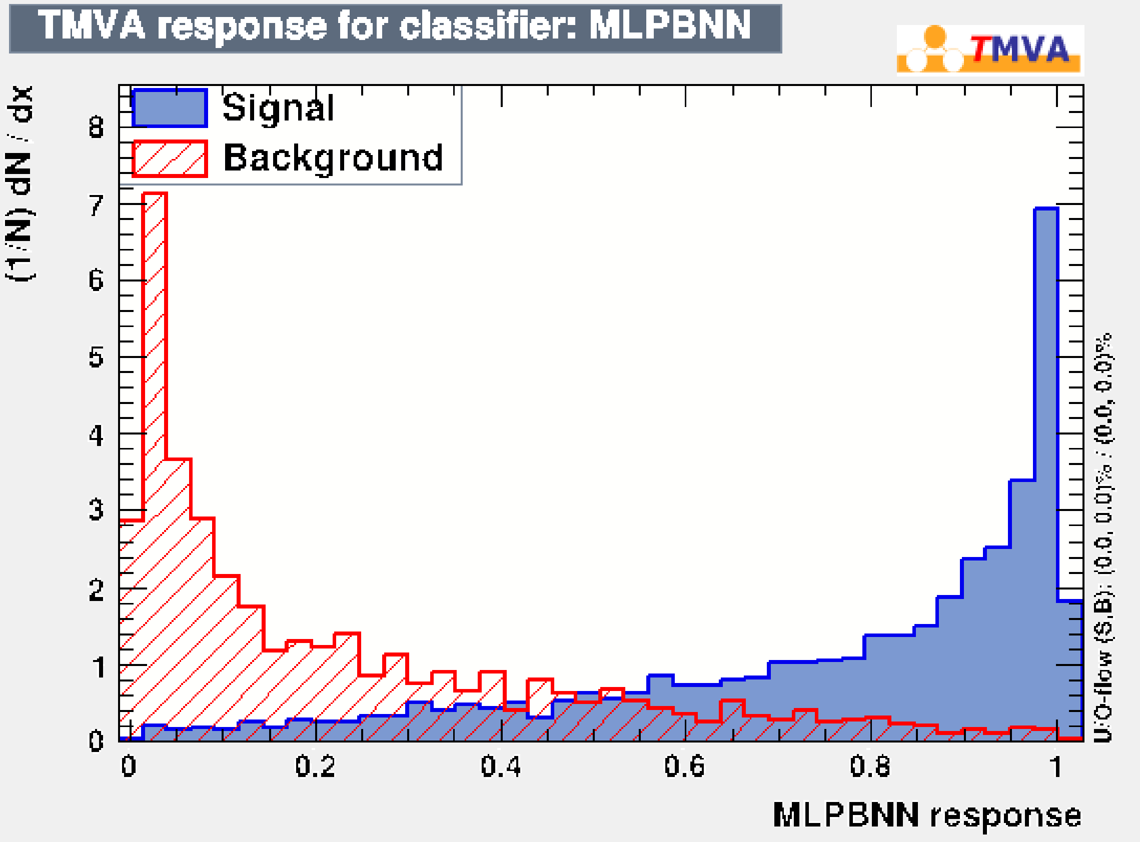
TMVA

TMVA is a ROOT-integrated toolkit for multivariate classification and regression analysis. TMVA performs the training, testing and performance evaluation of a large variety of multivariate methods.

supervised machine learning

1. Training phase:
train(build), test and evaluate classifiers using data samples with known signal and background events

2. Application phase: use to classify unknown data samples



Go to: `$ROOTSYS/bin/root/tmva/test`

run `root -l TMVAClassification.C`

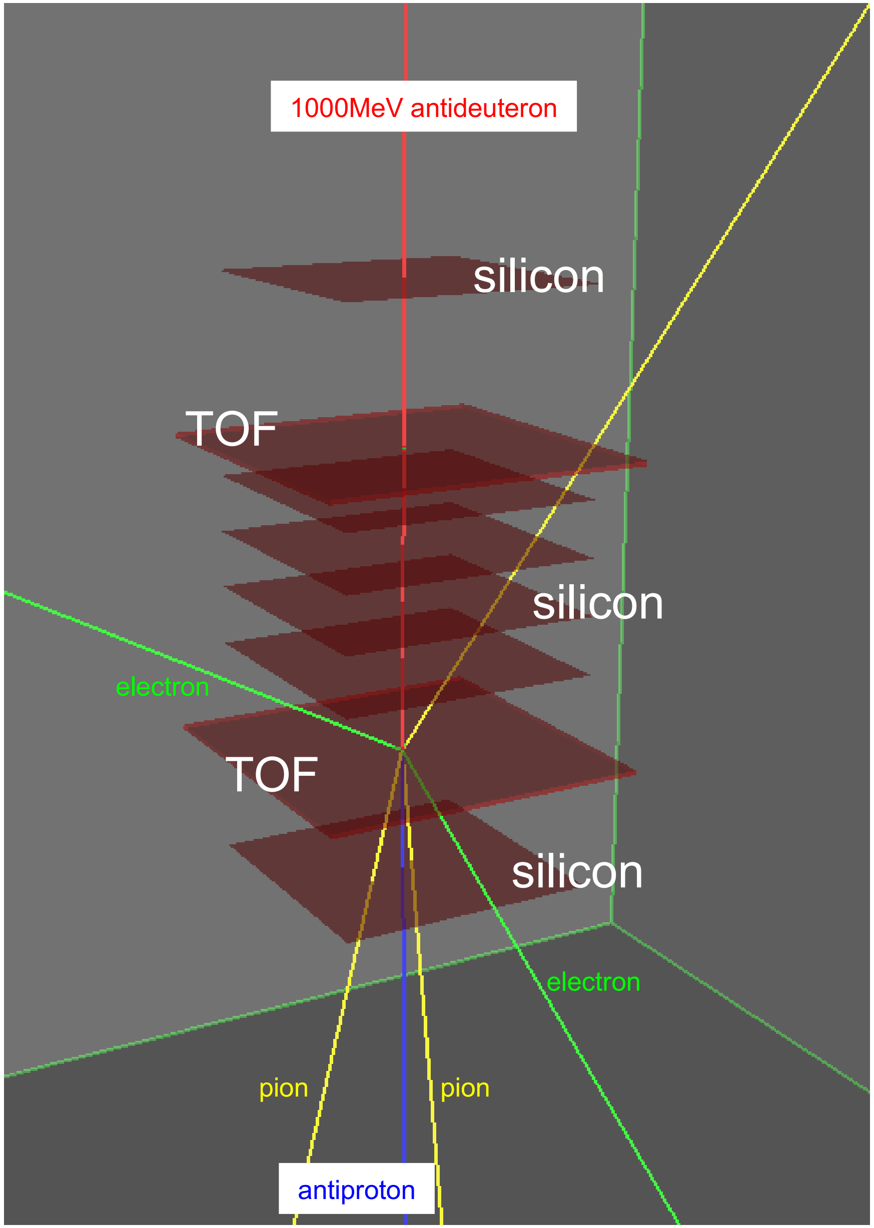
Integration to Geant4

Simulation of particle interactions with matter:

→ Geant4 is also based on C++
→ save simulation results as ROOT files

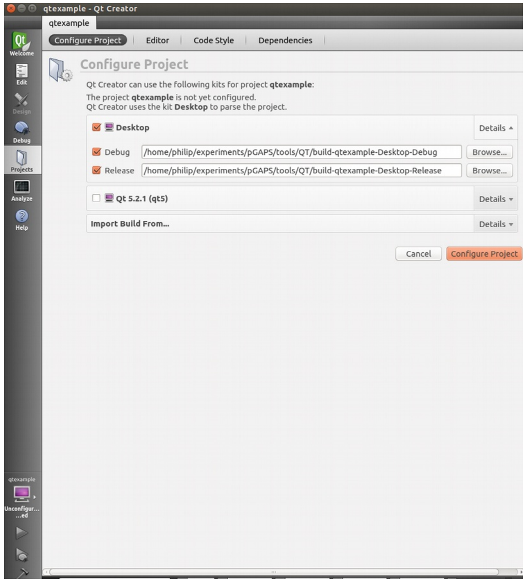
→ Add FindROOT.cmake to build directory
→ Add to CmakeLists.txt

```
# Find ROOT
find_package(ROOT)

# Add ROOTs header paths
include_directories($
{ROOT_INCLUDE_DIR})
```



QT

>rm -r Builds/ makefile qtexample.pro.user
>qmake-qt4
>qmake
>qtcreator qtexample.pro &
→ configure project

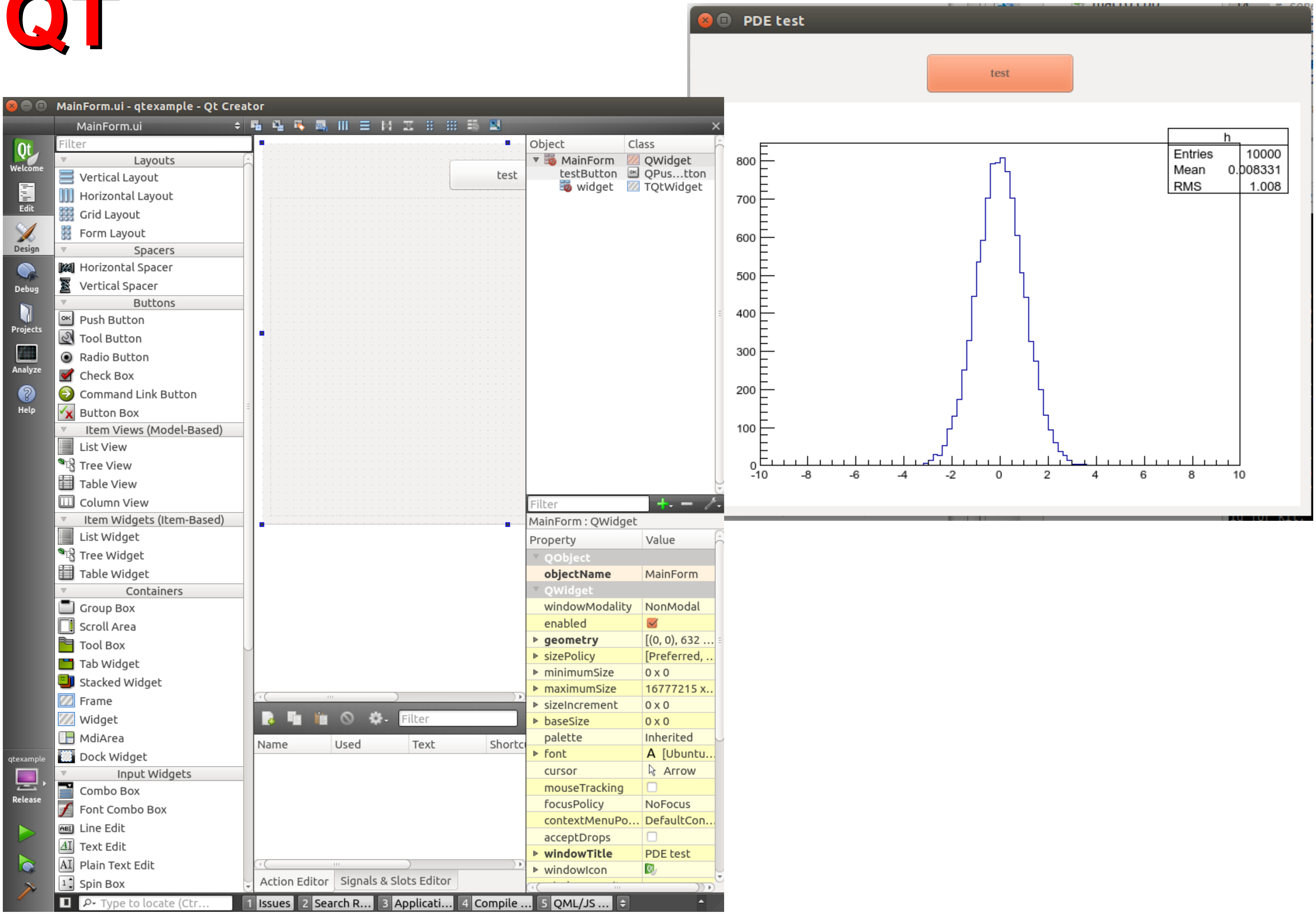



```
#include "MainWidget.h"

#include "TROOT.h"
#include "TCanvas.h"
#include "TH1.h"

MainWidget::MainWidget(QWidget* parent)
  : QDialog(parent)
{
  setupUi(this);

  h = new TH1I("h", "", 100, -10, 10);

  gROOT->Reset();
  m_canvas = gPad->GetCanvas();

  connect(testButton, SIGNAL(clicked()), this,
SLOT(testButtonClicked()));
}

MainWidget::~MainWidget()
{
}

void MainWidget::testButtonClicked()
{
h->FillRandom("gaus", 10000);
widget->cd();
h->Draw();
m_canvas->Modified();
m_canvas->Update();
}
```

QT - .pro file

```
CONFIG      += qt debug
TEMPLATE     = app

INCLUDEPATH += $(ROOTSYS)/include
LIBS        += $$system(root-config --cflags --libs) -lGQt

SOURCES     += main.cpp

HEADERS     += MainWidget.h
SOURCES     += MainWidget.cpp

FORMS       += MainForm.ui

# seperate source & build dirs
DESTDIR = ./Builds
OBJECTS_DIR = ./Builds/Temp
MOC_DIR = ./Builds/Temp
UI_DIR = ./Builds/Temp
RCC_DIR = ./Builds/Temp
```

QT


MainForm.ui - qtexample - Qt Creator
PDE test
test
h
Entries 10000
Mean 0.008331
RMS 1.008
Layouts
Vertical Layout
Horizontal Layout
Grid Layout
Form Layout
Spacers
Horizontal Spacer
Vertical Spacer
Buttons
Push Button
Tool Button
Radio Button
Check Box
Command Link Button
Button Box
Object
Class
MainForm
QWidget
testButton
QPus...tton
widget
TQtWidget
MainForm : QWidget
Property
Value
objectName
MainForm
windowTitle
PDE test
Action Editor
Signals & Slots Editor

QT online software example