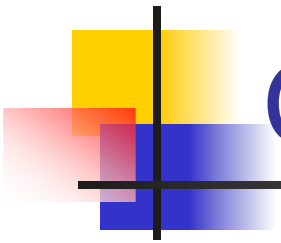
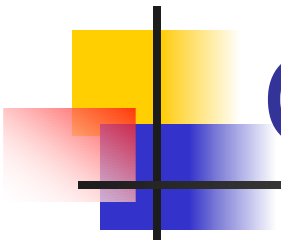


Generare ed istogrammare variabili

- Generare casualmente una variabile a partire da una certa distribuzione.
- Salvare le variabili in opportune strutture.
- Trattare i dati salvati in queste strutture.
- Costruire degli istogrammi.



Generare delle variabili

- Generiamo tre variabili a partire dalle seguenti distribuzioni:
  - Gaussiana.
  - Uniforme.
  - Polinomio di secondo grado.
- Classi usate:
  - TFile: gestione dei file root.
  - TNtuple e TTree: conservazione dei dati.

Generare delle variabili

- Listato:

  File: genera.cc

```
#include <TMath.h>
Double_t Gaus (Double_t* x, Double_t* par)
{
 return exp(-0.5*pow((x[0]-par[0])/par[1],2))/
        (par[1]*sqrt(2*TMath::Pi()));
}

Double_t X2 (Double_t* x, Double_t* par)
{
 return TMath::Abs(x[0]*x[0]*par[0]+
                   x[0]*par[1]+par[2]);
}
```

Generare delle variabili

```
void genera()
{
  gROOT->Reset();
  gRandom->SetSeed(0);

  const Int_t num = 1000;
  const Double_t minGaus = 1.19;
  const Double_t maxGaus = 1.25;
  const Double_t minX2 = -2.0;
  const Double_t maxX2 = 2.0;
  const Double_t parGaus[2] = {1.22,0.03};
  const Double_t parX2[3] = {2.2,0,2};
```

Generare delle variabili

```
// Dichiaro le distribuzioni
TF1* pdfGaus = new
 TF1("pdfGaus",Gaus,minGaus,maxGaus,2);
pdfGaus->SetParameters(parGaus);

TF1* pdfX2 = new TF1("pdfX2",X2,minX2,maxX2,3);
pdfX2->SetParameters(parX2);

// Dichiaro il file su cui salvare
TFile* f = (TFile*)gROOT->FindObject("genera.root");
if (f) f->Close();
f = new TFile ("genera.root","RECREATE",
               "Variabili generate casualmente");
```

Generare delle variabili

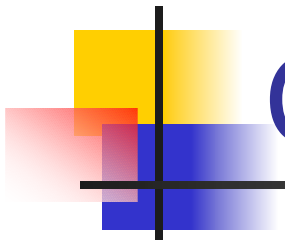
```
Int_t evt;

// Mi servo di una ntupla per salvare le variabili

TNtuple* nt = new TNtuple ("nt",
                "Variabili generate (ntupla)",
                "evt:rand:gaus1:gaus2:x2");

for (evt = 0; evt<num; evt++)
 nt->Fill(evt,gRandom->Rndm(),
        gRandom->Gaus(parGaus[0],parGaus[1]),
        pdfGaus->GetRandom(),pdfX2->GetRandom());
```

Generare delle variabili

```
TCanvas* c1 = new TCanvas("c1",
              "Variabili generate (ntupla)",
              160,30,700,500);
c1->Divide(2,2);

c1->cd(1);
nt->Draw("rand");
c1->cd(2);
nt->Draw("gaus1");
c1->cd(3);
nt->Draw("gaus2");
c1->cd(4);
nt->Draw("x2");
```

Generare delle variabili

```
// Mi servo di un albero per conservare le variabili

struct var_t
{
 Int_t evt;
 Float_t gaus1, gaus2, x2, rand;
} var;

TTree* tree = new TTree("tree",
       "Variabili generate (albero)");

tree->Branch("evt",&var.evt,"evt/I");
tree->Branch("gaus1",&var.gaus1,"gaus1/F");
tree->Branch("gaus2",&var.gaus2,"gaus2/F");
tree->Branch("x2",&var.x2,"x2/F");
tree->Branch("rand",&var.rand,"rand/F");
```



Generare delle variabili

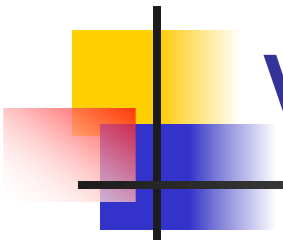
```
for (var.evt = 0; var.evt<num; var.evt++)
{
 var.gaus1 = gRandom->Gaus(parGaus[0],parGaus[1]);
 var.gaus2 = pdfGaus->GetRandom();
 var.x2 = pdfX2->GetRandom();
 var.rand = gRandom->Rndm();
 tree->Fill();
}

tree->Print();
```

Generare delle variabili

```
TCanvas* c2 = new TCanvas("c2",
      "Variabili generate (albero)",
      180,50,700,500);
c2->Divide(2,2);

c2->cd(1);
tree->Draw("rand");
c2->cd(2);
tree->Draw("gaus1");
c2->cd(3);
tree->Draw("gaus2");
c2->cd(4);
tree->Draw("x2");
```

Generare delle variabili

```
    // Salvo le variabili nel file
    f->Write();

    // Chiudo il file
    f->Close();

}
```



Visualizzare i dati salvati

- Visualizziamo le variabili generate precedentemente impiegando varie opzioni.

- Classi usate:
  - TCut: gestione dei tagli sulle variabili.

Visualizzare i dati salvati

- Listato:

  File: albero.cc

```
void albero()
{
 gROOT->Reset();
 TFile* f = (TFile*)gROOT->FindObject("genera.root");
 if (f) f->Close();
 f = new TFile ("genera.root");

 // Istogrammo le variabili dell'albero (o ntupla)
 TCanvas* c1 = new TCanvas("c1",
               "Istogrammi 1D",160,30,700,500);
 c1->Divide(2,2);
```

Visualizzare i dati salvati

```
c1->cd(1);
tree->Draw("sqrt(gaus1**3+1)");

c1->cd(2);
tree->Draw("gaus1+gaus2","rand<0.5 && evt<=200");

c1->cd(3);
TCut cut1 = "evt>200";
TCut cut2 = "evt<300";
TCut cut = cut1 && cut2;
tree->Draw("gaus1+gaus2",cut && "rand<0.5");
```

Visualizzare i dati salvati

```
c1->cd(4);
TH1F* h1 = new TH1F ("h1","Istogramma 1D",100,-2,2);
h1->SetFillColor(4);
h1->SetXTitle("Variabile indipendente");
h1->SetYTitle("Variabile dipendente");
tree->Draw("x2>>h1");
h1->SetTitleOffset(1.2,"X");
```

Visualizzare i dati salvati

```
TCanvas* c2 = new TCanvas("c2",
                "Istogrammi 2D e 3D",180,50,700,500);
c2->Divide(2,2);

c2->cd(1);
tree->Draw("gaus1:gaus2");

c2->cd(2);
TH2F* h2 = new TH2F ("h2","Istogramma 2D",
              100,1.17,1.27,100,1.1,1.4);
tree->Draw("gaus1:gaus2>>h2","rand<0.5");
```

Visualizzare i dati salvati

```
c2->cd(3);
tree->Draw("gaus1:gaus2","","lego");
/*
  TH2F* h2a = new TH2F ("h2a","Istogramma 2D",
                        20,1.17,1.27,20,1.1,1.4);
  tree->Draw("gaus1:gaus2>>h2a","","lego");
*/

c2->cd(4);
tree->Draw("gaus1:gaus2:x2");
```

Visualizzare i dati salvati

```
TCanvas* c3 = new TCanvas("c3","Altre opzioni",
                          200,70,700,500);
c3->Divide(1,2);
c3_1->Divide(2,1);

// sovrapposizione di due istogrammi
c3_1->cd(1);
TH1F* hs = new TH1F ("hs","Sovrapposizione",
                     100,1.1,1.35);
tree->Draw("gaus2>>hs");
hs->SetLineColor(2);
tree->Draw("gaus1","","same");
```

Visualizzare i dati salvati

```
// somma di due istogrammi
c3_1->cd(2);
TH1F* ha = new TH1F(*hs);
ha->SetName("ha");
ha->SetTitle("Somma");
tree->Draw("gaus2>>ha");
tree->Draw("gaus1>>+ha");

// selezione dei valori da istogrammare
c3_2->cd();
tree->Draw("gaus2","","",100,500);

}
```