```
void setStyle(){
  gROOT->SetStyle("Plain");
  gStyle->SetOptStat(112210);
  gStyle->SetOptFit(111);
  gStyle->SetPalette(57);
  gStyle->SetOptTitle(0);
}



void MonteCarlo() {
// funzione principale della macro



   Int_t nbins=1000;
//creazione istogrammi

   Double_t xmin=-20;
   Double_t xmax=20;


TH1F *h1 = new TH1F("h1","Gauss 1",nbins,xmin,xmax);
TH1F *h2 = new TH1F("h2","Gauss 2",nbins,xmin,xmax);
TH1F *h3 = new TH1F("h3","Sum 1+2",nbins,xmin,xmax);

Double_t x1,x2,x3;
for(Int_t i=0;i<1000000;i++){
  x1=gRandom->Gaus(-1,3);
  x2=gRandom->Gaus(1,4);
  x3=x1+x2;
  h1->Fill(x1);
  h2->Fill(x2);
  h3->Fill(x3);

}


TCanvas *cRandom = new TCanvas("cRandom","Test Somma Gaussiane");
cRandom->Divide(2,2);
//cosmetica: assi, colore, spessore linea,tipo di Marker...
h1->SetLineColor(kRed);
h2->SetLineColor(kBlue);
h3->SetLineColor(kMagenta);
h1->GetXaxis()->SetTitle("x1");
h2->GetXaxis()->SetTitle("x2");
h3->GetXaxis()->SetTitle("x3=x1+x2");

cRandom->cd(1);h1->Draw();
cRandom->cd(2);h2->Draw();
cRandom->cd(3);h3->Draw();
h3->Fit("gaus");
}
```

```
void MonteCarlo() {
// funzione principale della macro


   Int_t ngen=10000;
   Int_t nbins=100;

//creazione array di istogrammi
   TString names[4]={"test1","test2","test3","test4"};
   Double_t xmin[4]={1,0,0,1};
   Double_t xmax[4]={3,4,1,2};
   TH1F *h[4];
   for (Int_t i=0;i<4;i++){
   h[i] = new TH1F("h_"+names[i],"Uniform "+names[i],nbins,xmin[i],xmax[i]);
        h[i]->GetXaxis()->SetTitle("x");
   h[i]->GetYaxis()->SetTitleOffset(1.3);
   h[i]->GetYaxis()->SetTitle("Occorrenze");
   h[i]->SetFillColor(kBlue);
   h[i]->SetLineWidth(2);
   h[i]->SetTitleSize(0.05);
        h[i]->SetMinimum(0);
   }

//riempimento array di istogrammi (Fill)
   for (Int_t i=0;i<4;i++){
    for(Int_t j=0;j<ngen;j++){
      Double_t x=gRandom->Uniform(xmin[i],xmax[i]);
           h[i]->Fill(x);
         }
   }


//Questa è una Canvas, la finestra grafica, che dividerò in 4 parti

 TCanvas *cRandom = new TCanvas("cRandom","Test Distribuzione Uniforme");
 cRandom->Divide(2,2);

//cosmetica: assi, colore, spessore linea,tipo di Marker...

 for (Int_t i=0;i<4;i++){
  cRandom->cd(i+1);
  h[i]->Draw();
  cout << "*------------------------------------------------------------*" <<endl;
  cout << "Occorrenze Totali: " <<h[i]->GetEntries() <<endl;
  cout << "Media dell'istogramma: " <<h[i]->GetMean() << " +/- " <<h[i]->GetMeanError()<<endl;
  cout << "RMS dell'istogramma: " <<h[i]->GetRMS() << " +/- " <<h[i]->GetRMSError()<<endl;
  cout << "Errore previsto media dell'istogramma: " << h[i]->GetRMS()/sqrt(ngen) <<endl;
  cout << "Errore previsto RMS dell'istogramma: " << h[i]->GetRMS()/sqrt(2*ngen) <<endl;
 }
}
```

```
void setStyle(){
  gROOT->SetStyle("Plain");
  gStyle->SetOptStat(1111);
  gStyle->SetOptFit(111);
  gStyle->SetPalette(57);
  gStyle->SetOptTitle(0);
}



void MonteCarlo() {
// funzione principale della macro


   Int_t ngen=100000;

   Int_t mean[4]={5,10,30,50};
   Int_t nbins[4]={20,30,60,100};
//creazione istogramma
   TString names[4]={"test1","test2","test3","test4"};
   Double_t xmin[4]={0,0,0,0};
   Double_t xmax[4]={20,30,60,100};
TH1F *h[4];
   for (Int_t i=0;i<4;i++){
   h[i] = new TH1F("h_"+names[i],"Binomial "+names[i],nbins[i],xmin[i],xmax[i]);
      h[i]->GetXaxis()->SetTitle("x");
   h[i]->GetYaxis()->SetTitleOffset(1.3);
   h[i]->GetYaxis()->SetTitle("Occorrenze");
   h[i]->SetFillColor(kBlue);
   h[i]->SetLineWidth(2);
   h[i]->SetMarkerStyle(4);
h[i]->SetMinimum(0);
    for(Int_t j=0;j<ngen;j++){
      Double_t x=gRandom->Poisson(mean[i]);
         h[i]->Fill(x);
}
}


//Questa è una Canvas, la finestra grafica....
 TCanvas *cRandom = new TCanvas("cRandom","Test Distribuzione Uniforme");
 cRandom->Divide(2,2);
//cosmetica: assi, colore, spessore linea,tipo di Marker...
 for (Int_t i=0;i<4;i++){
  cRandom->cd(i+1);
  h[i]->Draw();
   cout << "*------------------------------------------------------------*" <<endl;

 cout << "Occorrenze Totali: " <<h[i]->GetEntries() <<endl;
 cout << "Media dell'istogramma: " <<h[i]->GetMean() << " +/- " <<h[i]->GetMeanError()<<endl;
 cout << "RMS dell'istogramma: " <<h[i]->GetRMS() << " +/- " <<h[i]->GetRMSError()<<endl;
}
}
```

```
void setStyle(){
  gROOT->SetStyle("Plain");
  gStyle->SetOptStat(1111);
  gStyle->SetOptFit(111);
  gStyle->SetPalette(57);
  gStyle->SetOptTitle(0);
}



void MonteCarlo() {
// funzione principale della macro



  Int_t nbins=1000;
//creazione istogrammi

  Double_t xmin=-1;
  Double_t xmax=11;


TH1F *h1 = new TH1F("h1","sum 1",nbins,xmin,xmax);
TH1F *h2 = new TH1F("h2","sum 2",nbins,xmin,xmax);
TH1F *h3 = new TH1F("h3","sum 3",nbins,xmin,xmax);
TH1F *h4 = new TH1F("h4","sum 10",nbins,xmin,xmax);
Double_t x[12];
for(Int_t i=0;i<1000000;i++){
  for(Int_t j=0;j<12;j++)x[j]=gRandom->Uniform(0,1);
  h1->Fill(x[0]);
  h2->Fill(x[0]+x[1]);
  h3->Fill(x[0]+x[1]+x[2]);
  Double_t sum=0;
  for(Int_t j=0;j<12;j++)sum+=x[j];
  h4->Fill(sum);
}


TCanvas *cRandom = new TCanvas("cRandom","Test Somma di N variabili");
cRandom->Divide(2,2);

h1->SetLineColor(kRed);
h2->SetLineColor(kBlue);
h3->SetLineColor(kMagenta);
h1->GetXaxis()->SetTitle("x1");
h2->GetXaxis()->SetTitle("x1+x2");
h3->GetXaxis()->SetTitle("x1+x2+x3");
h4->GetXaxis()->SetTitle("x1+...+x12");
cRandom->cd(1);h1->Draw();
cRandom->cd(2);h2->Draw();
cRandom->cd(3);h3->Draw();
cRandom->cd(4);h4->Draw();
h4->Fit("gaus");
}
```

```
void setStyle(){
  gROOT->SetStyle("Plain");
  gStyle->SetOptStat(1111);
  gStyle->SetOptFit(111);
  gStyle->SetPalette(57);
  gStyle->SetOptTitle(0);
}


Double_t myfunction(Double_t *x, Double_t *par){
Double_t var=x[0];
return par[2]*TMath::CauchyDist(var,par[0],par[1]);

}

void MonteCarlo() {
// funzione principale della macro



   Int_t nbins=1000;
//creazione istogrammi

   Double_t xmin=-20;
   Double_t xmax=20;


TH1F *h1 = new TH1F("h1","Angolo",nbins,-TMath::Pi()/2.,TMath::Pi()/2.);
TH1F *h2 = new TH1F("h2","variabile X",nbins,-10,10);

Double_t Theta,X;
for(Int_t j=0;j<1000000;j++){
Theta=gRandom->Uniform(-TMath::Pi()/2.,TMath::Pi()/2.);
h1->Fill(Theta);
h2->Fill(TMath::Tan(Theta));
 }


TCanvas *cRandom = new TCanvas("cRandom","Test Somma di Gaussiane");
cRandom->Divide(2,2);

h1->SetLineColor(kRed);
h1->SetMinimum(0);
h2->SetLineColor(kBlue);

h1->GetXaxis()->SetTitle("Theta");
h2->GetXaxis()->SetTitle("Tan(Theta)");

cRandom->cd(1);h1->Draw();
cRandom->cd(2);h2->DrawClone();
TF1 *f = new TF1("f",myfunction,-10,10,3);
f->SetParameters(0,1, 6000);
cRandom->cd(3);
//f->Draw();
h2->Fit("f");
}
```

```
void setStyle(){
  gROOT->SetStyle("Plain");
  gStyle->SetOptStat(1111);
  gStyle->SetOptFit(111);
  gStyle->SetPalette(57);
  gStyle->SetOptTitle(0);
}



void MonteCarlo() {
// funzione principale della macro


   Int_t ngen=100000;

   Int_t ntot[4]={10,100,1000,10000};
   Int_t nbins[4]={20,20,20,20};
   Double_t prob=0.;
//creazione istogramma
   TString names[4]={"test1","test2","test3","test4"};
   Double_t xmin[4]={0,0,0,0};
   Double_t xmax[4]={20,20,20,20};
TH1F *h[4];
   for (Int_t i=0;i<4;i++){
   h[i] = new TH1F("h_"+names[i],"Binomial "+names[i],nbins[i],xmin[i],xmax[i]);
       h[i]->GetXaxis()->SetTitle("x");
   h[i]->GetYaxis()->SetTitleOffset(1.3);
   h[i]->GetYaxis()->SetTitle("Occorrenze");
   h[i]->SetFillColor(kBlue);
   h[i]->SetLineWidth(2);
   h[i]->SetMarkerStyle(4);
h[i]->SetMinimum(0);
    for(Int_t j=0;j<ngen;j++){
                prob=5./ntot[i];
      Double_t x=gRandom->Binomial(ntot[i],prob);
           h[i]->Fill(x);
}
}


//Questa è una Canvas, la finestra grafica....
 TCanvas *cRandom = new TCanvas("cRandom","Test Distribuzione Uniforme");
 cRandom->Divide(2,2);
//cosmetica: assi, colore, spessore linea,tipo di Marker...
 for (Int_t i=0;i<4;i++){
  cRandom->cd(i+1);
  h[i]->Draw();
   cout << "*------------------------------------------------------------*" <<endl;

 cout << "Occorrenze Totali: " <<h[i]->GetEntries() <<endl;
 cout << "Media dell'istogramma: " <<h[i]->GetMean() << " +/- " <<h[i]->GetMeanError()<<endl;
 cout << "RMS dell'istogramma: " <<h[i]->GetRMS() << " +/- " <<h[i]->GetRMSError()<<endl;
}
}
```

```
void setStyle(){
  gROOT->SetStyle("Plain");
  gStyle->SetOptStat(1111);
  gStyle->SetOptFit(111);
  gStyle->SetPalette(57);
  gStyle->SetOptTitle(0);
}



void MonteCarlo() {
// funzione principale della macro


   Int_t ngen=100000;

   Int_t ntot[4]={5,10,50,100};
   Int_t nbins[4]={5,10,50,100};
   Double_t prob=0.3;
//creazione istogramma
   TString names[4]={"test1","test2","test3","test4"};
   Double_t xmin[4]={0,0,0,0};
   Double_t xmax[4]={5,10,50,100};
TH1F *h[4];

   for (Int_t i=0;i<4;i++){
   h[i] = new TH1F("h_"+names[i],"Binomial "+names[i],nbins[i],xmin[i],xmax[i]);
//cosmetica: assi, colore, spessore linea,tipo di Marker...
      h[i]->GetXaxis()->SetTitle("x");
   h[i]->GetYaxis()->SetTitleOffset(1.3);
   h[i]->GetYaxis()->SetTitle("Occorrenze");
   h[i]->SetFillColor(kBlue);
   h[i]->SetLineWidth(2);
   h[i]->SetMarkerStyle(4);
      h[i]->SetMinimum(0);
    for(Int_t j=0;j<ngen;j++){
      Double_t x=gRandom->Binomial(ntot[i],prob);
         h[i]->Fill(x);
}
}


//Questa è una Canvas, la finestra grafica....
 TCanvas *cRandom = new TCanvas("cRandom","Test Distribuzione Uniforme");
 cRandom->Divide(2,2);

 for (Int_t i=0;i<4;i++){
  cRandom->cd(i+1);
  h[i]->Draw();
   cout << "*-------------------------------------------------------*" <<endl;

 cout << "Occorrenze Totali: " <<h[i]->GetEntries() <<endl;
 cout << "Media dell'istogramma: " <<h[i]->GetMean() << " +/- " <<h[i]->GetMeanError()<<endl;
 cout << "RMS dell'istogramma: " <<h[i]->GetRMS() << " +/- " <<h[i]->GetRMSError()<<endl;
}
}
```

```
void setStyle(){
  gROOT->SetStyle("Plain");
  gStyle->SetOptStat(1111);
  gStyle->SetOptFit(0);
  gStyle->SetPalette(57);
  gStyle->SetOptTitle(0);
}

void MonteCarlo() {
// funzione principale della macro
    const Int_t npoints=5;
    Int_t nbins=100;

//creazione istogramma

    Double_t xmin=0;
    Double_t xmax=1;
    Double_t x[npoints],y[npoints],ey[npoints];

    TH1F *h = new TH1F("h","Uniform",nbins,xmin,xmax);

for(Int_t i=0;i<npoints;i++){
 for(Int_t j=0;j<pow(10,i+1);j++){
     Double_t var=gRandom->Uniform(xmin,xmax);
                h->Fill(var);
 }
 x[i]=log10(h->GetEntries());
 y[i]=h->GetMean(); //media in un array
 ey[i]=h->GetMeanError(); //errore media in un array
 h->Reset(); //reset dell'istogramma per l'iterazione successiva
}

 // i risultati in due Grafici
TCanvas *cRandom = new TCanvas("cRandom","Test Distribuzione Uniforme");
  cRandom->Divide(1,2);

  cRandom->cd(1);
  TGraphErrors*graph=new TGraphErrors(npoints,x,y,0,ey);
  graph->SetTitle("Media della distribuzione U(0,1); log10(N); media");
  graph->SetMinimum(0);
  graph->SetMaximum(1);
  graph->SetMarkerStyle(kCircle);
  graph->SetMarkerColor(kRed);
  graph->Draw("APE");
  graph->GetXaxis()->SetTitleSize(0.05);
  graph->GetYaxis()->SetTitleSize(0.05);
  graph->GetXaxis()->SetLabelSize(0.05);
  graph->GetYaxis()->SetLabelSize(0.05);
  cRandom->cd(2);
```

```
 TGraph *graphMeanErr=new TGraph(npoints,x,ey);
 graphMeanErr->SetTitle("Media della distribuzione U(0,1); log10(N); Errore sulla media");
 graphMeanErr->SetMinimum(0);
 graphMeanErr->SetMaximum(0.2);
 graphMeanErr->SetMarkerStyle(kCircle);
 graphMeanErr->SetMarkerColor(kRed);
 graphMeanErr->GetXaxis()->SetTitleSize(0.05);
 graphMeanErr->GetYaxis()->SetTitleSize(0.05);
 graphMeanErr->GetXaxis()->SetLabelSize(0.05);
 graphMeanErr->GetYaxis()->SetLabelSize(0.05);
 graphMeanErr->Draw("APE");

/Verifica della dipendenza attesa (1/sqrt(n))

 TF1 *f=new TF1("f","[0]/sqrt(10**x)",0,5);
 f->SetLineColor(kRed);
 graphMeanErr->Fit("f");
```