Table of Contents

AtlasJapanSoftwareTutorial16DecROOT

ROOT
ROOT 多く ラ ブラリ(用途ご &#125
ROOT ラス名 普通大文字 Tから&#

使 方 大きく分けて二つあ&#1

*
ROOT パッケ ジ 含まれ プリ (Cling

*
自分 マ やプ グラム でROOT &#1252

どちらでも基本的な使 方 &#2

Cling ベ 分布 見 きなど 使 ま

でよく使 ラス

際 行 処理 純 読み込&#1241
最終結果 分岐比や質量な&#12
保存時 構造 してROOTで
ツリ (`TTree`) 使 ま
ま 図 書く際 基本的 ヒス &#12
TH2)かグラフ(TGraph, TGraphErrors) 使 ま

ATLASで 生 セ ラルプ セスでDAOD

環境設定

今回 講習 login.icepp.jpで行 ま sshでlogin&#1237

ATLASで localSetupROOT等でROOT 使用 必要な&
PATH, LD_LIBRARY_PATH
(Macで DYLD_LIBRARY_PATH) 三つ 設定してく

ラ ティブセッ

ROOT ラ ティブセッ (Cling) 通常 &#2019
`root` プして始 ま
以下 例 あ よ ほ んど C++ コ &#1248
それ 加えて累乗 よ な拡張&
特殊コマ ド ドッ で始まり&
よく使 メモリ上 あ オブ&#1247
通常 .qで終了しま が プ グ&#12521

```
$ root
root [] cout<<"Hello world"<<endl;
root [] 3 * 2
root [] cos(TMath::Pi())
root [] 3^2
root [] for (int i = 0; i < 10; i++) {
```

```
root (cont'ed, cancel with .@) [] cout<<i<<endl;
root (cont'ed, cancel with .@) [] }
root [] TTree tmp("tmp", "tmp")
root [] .ls
root [] .help
root [] .q
```

便利な機能

Cling ヒス リ 機能があ で 矢印&

ま コマ ド 補完機能もあり&

```
root [] TGraphEr
```

後でTab 打つ Clingが知って ラ&#1247

さら 下 よ な状態でTab 打つ

```
root [] TGraphErrors a(
```

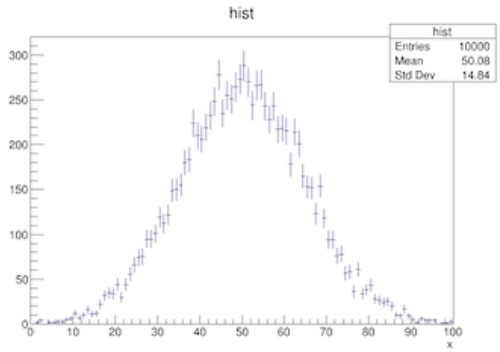
こ ラス コ ス ラ 引数 知 こ &

ラ ティブセッ 起動 終&#20102

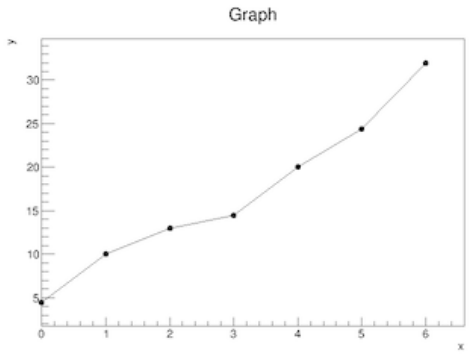
ヒス グラム &#124

ヒス グラム 一次元以上 変&
物理 で ベ 数だけで なく&#354
それ 対して グラフ 二次元&#

ヒス グラム 例



グラフ 例



ヒス グラム

ROOT 基本的なヒス グラム ラ&#1
TH1S,
TH2D等 名前がつ て ま 3文字目&
4文字目 ビ 身(普通 ベ 数) 型
short, I: int, F:float, D:double
等がありま ベ 重み つけ &#12

それで 一次元 ヒス グラム &

```
$ root
root [] TH1F hist("hist", "Histogram;x;number of events", 100, -10, 10)
root [] for (int i = 0; i < 10000; i++) { hist.Fill(gRandom->Gaus(0, 7)); }
root [] TCanvas c1("c1", "c1")
root [] hist.Draw()
```

ここで 一次元 ヒス グラム
TH1F 引数 順 こ オブジェ 名&#21069
ル で ;
で区切られ 二番目 x軸 三番&
gRandom ROOT でグ バル 定義されて

(他 もgROOT, gSystem,
gPad等 グ バルな変数があり &#235
ヒス グラム 表示 キャ バ&#12

ヒス グラム 情報 得てみま

```
root [] hist.GetEntries()
root [] hist.GetMean()
root [] hist.GetRMS()
root [] hist.GetBinContent(50)
root [] hist.GetBinError(50)
root [] hist.GetBinContent(0)
root [] hist.GetBinContent(101)
```

ここで histが持って 情報(エ &#1
そ 次 例 して50番目 ビ 高さ &#1
それで正しくな 場合 自分&#1
ヒス グラム 最初 最後 ビ &#2930

横 それま が ROOTで コ ド ポ &#1250
UInt_t,
Float_t等 基本型 定義して ま
基本型 フルリス :
https://root.cern.ch/root/html528/ListOfTypes.html

で ヒス グラム 見 目 変え&#1239

```
root [] hist.SetLineColor(kRed)
root [] hist.SetLineWidth(2)
root [] hist.SetLineStyle(1)
root [] hist.SetFillColor(kRed)
root [] hist.Draw("")
root [] hist.Draw("e")
```

軸 ル 後から変え こ もで&#123

```
root [] hist.GetXaxis()->SetTitle("#theta")
```

上 例 よ #
使 LaTeX コマ ド 一部 使 こ が&#123

今回 Draw 時 eオプ つけ こ で&#12

同じよ なこ グラフィカル&
ヒス グラムや軸上 右 リッ
例えば x軸 右 リッ し SetRangeUser 選&#
元 戻 UnZoom 選びま

ヒス グラムが選ばれて 時 &

で ヒス グラム フィッ して

```
root [] hist.Fit("gausn", "", "", -5, 5)
root [] hist.GetFunction("gausn")->GetParameter(0)
root [] hist.GetFunction("gausn")->GetParameter(1)
root [] hist.GetFunction("gausn")->GetParameter(2)
```

ここで　正規化されガウス&#
gausn
正規化されガウス分布で&#
sigmaで
正規化　必要がな場合 gaus
最初パラメ　がmeanでy軸方&#2152

ATLASで通常統計情報プッ　表&

```
root [] gStyle->SetOptStat(1)
root [] gStyle->SetOptFit(1)
root [] hist.Draw()
```

y軸対数表示　場合

```
root [] c1.SetLogy(kTRUE)
root [] hist.Draw()
```

しま

横　それまがROOTで定数kから&
kFALSE 書かれま　(true/false 使っても大&#

キャバスからView->Toolbar リッし&#123
同じこ　コマドラ　でTLineやTLatex &#1

キャバスSaveAs関数で保存　こ&

```
root [] c1.SaveAs("canvas.eps")
root [] c1.SaveAs("canvas.pdf")
root [] c1.SaveAs("canvas.png")
root [] c1.SaveAs("canvas.C")
```

ファル形式拡張子よって&
あでプッ　変更(色や線太&#1237

キャバス複数パッド分割
ここで　縦横1x2 分割してみ&#12414
他オブジェ　Draw 前cd こで描&#30
パッド　ッス1始まりで　(0 キ&

```
root [] c1.Clear()
root [] c1.Divide(2,1)
root [] c1.cd(1)
```

時　ベ　重み付け場合が&#12
例えばMC ベ　フィル時そ&#1250
そ場合　Fill関数二つ目引数&#

```
root [] hist.Reset()
root [] hist.Fill(2.5, 1.5)
root [] hist.Draw()
```

一つビ　1.5が入って思ま
ま　Fillし後　例えば積分ルミ&#124

```
root [] hist.Scale(36500)
root [] hist.Draw()
```

しま

次 二次元 ヒス グラム 作っ&

```
root [] c1.cd(2)
root [] TH2F hist2D("hist2D", "2D histogram", 100, -10, 10, 100, -10, 10)
root [] Double_t x,y;
root [] for (int i = 0; i < 100000; i++) { x = gRandom->Uniform(-10,10); y = x/10;  hist2D.Fill(x
root [] hist2D.Draw("lego")
root [] hist2D.Draw("colz")
```

コ ス ラ で 1次元 場合 加え&#12
重み付け 場合 三番目 引数
一番目 Draw レゴプ ッ で z軸 含
二番目 Drawで z軸 カラ コ ドで&#

2次元分布 フィッ TProfileが使&#1236
(他 もFitSlicesY 使 方法もありま が

```
root [] TProfile *prof = hist2D.ProfileX("prof")
root [] prof->SetLineColor(kBlue)
root [] prof->Draw("same")
root [] prof->Fit("pol1")
```

ここで 二次元 分布 プ ファ

グラフ

グラフ 基本的な ラス `TGraph`
で が ほ んど場合 エラ バ &#2018
`TGraphErrors` 使 ま

```
root [] c1.Clear()
root [] Double_t xm[10] = {0.29, 0.86, 2.23, 2.99, 4.24, 4.98, 5.72, 6.97, 8.00, 8.87}
root [] Double_t xe[10] = {0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3}
root [] Double_t ym[10] = {1.30, 0.27, 4.08, 8.82, 14.82, 25.87, 36.40, 50.48, 64.52, 81.53}
root [] Double_t ye[10] = {1, 1, 1, 1, 1, 1, 1, 1, 1, 1}
root [] TGraphErrors g(10, xm, ym, xe, ye)
root [] g.SetMarkerStyle(7)
root [] g.Draw("apl")
```

こ 時 オプ 意味

* a:
グラフ 内容 応じて自動的 &
(二個目以後 グラフ Draw 時 a &#22806

* p: 点 表示

* l: 点 点 間 直線でつなぐ

で

グラフ そ ままフィッ でき&

```
root [] g.Fit("pol2")
```

ツリ

TTree わゆ n-tuple (複数 値 組み)
列 格納 ラスで 保存 広&
TTree
基本的な型 変数だけで な&
vector
や ラス等 オブジェ も格納
(ROOT TTree 別 TNtuple
ラスもありま が 私が知 &#384

それで 実際 ツリ 作ってみ
実際 で ツリ 大きさが何&#3033
そ 場合 ツリ RAM上 保持 こ が
それ 避け ツリ 作 前 TFile
作ってくださ これ よりTree &#19
(あらかじ サ ズが大きくな&#

```
root [] Int_t EventNumber
root [] Double_t METPt
root [] Double_t METEta
root [] vector<Double_t> muonPt
root [] vector<Double_t> muonEta
root [] vector<Double_t> MSTrackPt
root [] vector<Double_t> MSTrackEta
root [] TFile *file = TFile::Open("tree.root", "RECREATE")
root [] TTree tree("tree", "tree")
root [] tree.Branch("EventNumber", &EventNumber, "EventNumber/I")
root [] tree.Branch("METPt", &METPt, "METPt/D")
root [] tree.Branch("METEta", &METEta, "METEta/D")
root [] tree.Branch("muonPt", &muonPt)
root [] tree.Branch("muonEta", &muonEta)
root [] tree.Branch("MSTrackPt", &MSTrackPt)
root [] tree.Branch("MSTrackEta", &MSTrackEta)
root [] for(Int_t i=0;i<100;i++){
root (cont'ed, cancel with .@) [] EventNumber=i;
root (cont'ed, cancel with .@) [] METPt=gRandom->Uniform(20,500);
root (cont'ed, cancel with .@) [] METEta=gRandom->Uniform(-2,2);
root (cont'ed, cancel with .@) [] muonPt.clear();
root (cont'ed, cancel with .@) [] muonEta.clear();
root (cont'ed, cancel with .@) [] MSTrackPt.clear();
root (cont'ed, cancel with .@) [] MSTrackEta.clear();
root (cont'ed, cancel with .@) [] Int_t nmuon = gRandom->Poisson(10);
root (cont'ed, cancel with .@) [] for (Int_t imuon = 0; imuon < nmuon; imuon++) { muonPt.push_bac
root (cont'ed, cancel with .@) [] for (Int_t imuon = 0; imuon < nmuon; imuon++) { muonEta.push_ba
root (cont'ed, cancel with .@) [] Int_t nMSTrack = gRandom->Poisson(10);
root (cont'ed, cancel with .@) [] for (Int_t iMSTrack = 0; iMSTrack < nMSTrack; iMSTrack++) { MST
root (cont'ed, cancel with .@) [] for (Int_t iMSTrack = 0; iMSTrack < nMSTrack; iMSTrack++) { MST
root (cont'ed, cancel with .@) [] tree.Fill();
root (cont'ed, cancel with .@) [] }
root [] tree.Write()
root [] hist.Write()
root [] hist2D.Write()
root [] g.Write("graph")
root [] file->Close()
root [] delete file
root [] .q
```

上 例で EventNumber, MET, muon,
MSTrack 100 ベ 分ツリ 書 て保存し&#
MET 各 ベ 一つしかありませ&#

ツリ があ 変数 格納 場所 &#1250
基本的な型 場合 vector 場合で &#12
色々な型でブラ チ 作 こ が&
ツリ Fillが呼ばれ そ 時 変数 &
最後 ツリ ファ ル 書 て フ&#124
つ で ヒス グラム グラフも&

TFile 開く際 "RECREATE"オプ 使 同じ&#2
上書きせず 新し オブジェ
オプ なし 場合 "READ"オプ で&#382
ファ ル Close まで 完全 ィス &#2636

次 今書 ツリ 読み込んで&#1250

もし どこかで失敗して ま&#1
/home/sawada/public/tutorial16/tree.root 使ってくださ

```
$ root tree.root
root [] .ls
root [] tree->Print()
root [] tree->Show(0)
root [] tree->Draw("METPt")
```

rootコマ ド 後 ファ ル名 指定 &#
マ やプ グラムで こ よ 自&#21

```
TTree *tree = dynamic_cast<TTree*>(_file0->Get("tree"));
```

よ 必要がありま

ツリ プリ ツリ 情報 ブラ &#1
ツリ Show関数 一つ ベ 内容 &#3492
ツリ でプ ッ 書く際 上 例 &#12
こ 時表示され ヒス グラム h
こ ヒス グラム 後で再利用&#

```
root [] TH1F frame("frame", "frame", 50, 20, 500)
root [] tree->Draw("METPt>>frame")
```

2次元(や3次元) プ ッ 変数 コ :&

```
root [] tree->SetMarkerStyle(8)
root [] tree->Draw("METPt:EventNumber")
```

2次元 場合 最初 変数がy軸方
なお3次元 場合 左から順 x,y,z &#1

全 ベ プ ッ せず ベ 選ぶ&#22580

```
root [] tree->Draw("METEta:EventNumber", "")
root [] tree->SetMarkerColor(kRed)
root [] tree->Draw("METEta:EventNumber", "fabs(METEta)<1&&EventNumber>50", "same")
```

よ 引数 カッ 書きま カッ C+

同様 ミュ オ つ てもPt 分布 Draw
こ 場合 条件 満 ミュ オ 全&#123

```
root [] tree->Draw("muonPt", "fabs(muonEta)<1&&EventNumber>50")
```

あえて最初 ミュ オ だけで&#

```
root [] tree->Draw("muonPt[0]", "fabs(muonEta[0])<1&&EventNumber>50")
```

よ できま

次 よ な間違え よくあ も な
MSTrack
(ミュ オ スペ メ ） ラッ Pt&#1236
(実際 もっ 複雑なカッ で&#38291

```
root [] tree->Draw("muonPt", "MSTrackPt>200")
```

ROOTで配列やvector型 ッ スなし&#123
`MSTrackPT[5]>200` であれば `muonPt[5]`
がDrawされま
しかし 実際 内部検出器も&#
`muonPt` `MSTrackPt`
対応させ 方法 知らな で Draw&#

少し複雑なカッ 伴 プ ッ &#2031

```
root [] Int_t EventNumber
root [] Double_t METPt
root [] Double_t METEta
root [] vector<Double_t> *muonPt = 0
root [] vector<Double_t> *muonEta = 0
root [] vector<Double_t> *MSTrackPt = 0
root [] vector<Double_t> *MSTrackEta = 0
root [] tree->SetBranchAddress("EventNumber", &EventNumber)
root [] tree->SetBranchAddress("METPt", &METPt)
root [] tree->SetBranchAddress("METEta", &METEta)
root [] tree->SetBranchAddress("muonPt", &muonPt)
root [] tree->SetBranchAddress("muonEta", &muonEta)
root [] tree->SetBranchAddress("MSTrackPt", &MSTrackPt)
root [] tree->SetBranchAddress("MSTrackEta", &MSTrackEta)
root [] TH1F hmupt("hmupt", "Muon Pt", 100, 0, 500)
root [] for(Int_t i=0;i<tree->GetEntries();i++){
root (cont'ed, cancel with .@) [] tree->GetEntry(i);
root (cont'ed, cancel with .@) [] Int_t nmuon = muonPt->size();
root (cont'ed, cancel with .@) [] for (Int_t imuon = 0; imuon < nmuon; imuon++) { hmupt.Fill((*mu
root (cont'ed, cancel with .@) [] }
```

上 コ ドで ブラ チ 対応 カ&#1
例で 純 ミュ オ Pt ヒス グラ&#
例えば 実際 物理 で ミュ &#124

参考:
くさん ブラ チ 読む時 自分

```
root [] tree.MakeClass("readtree")
```

マ

決まっ 処理 繰り返し行っ &#

もっ も なマ Clingで コマ ド
{} 入れ も で 以下 コ ド macro.C
ファ ル 書 てくださ

入力が大変な人 こ ファ ル &

/home/sawada/public/tutorial16/macro.C

```
{
   TFile *file = TFile::Open("tree.root");
   TTree *tree = dynamic_cast<TTree*>(file->Get("tree"));
   Int_t EventNumber;
   Double_t METPt;
   Double_t METEta;
   vector<Double_t> *muonPt = 0;
   vector<Double_t> *muonEta = 0;
   vector<Double_t> *MSTrackPt = 0;
   vector<Double_t> *MSTrackEta = 0;
   tree->SetBranchAddress("EventNumber", &EventNumber);
   tree->SetBranchAddress("METPt", &METPt);
   tree->SetBranchAddress("METEta", &METEta);
   tree->SetBranchAddress("muonPt", &muonPt);
   tree->SetBranchAddress("muonEta", &muonEta);
   tree->SetBranchAddress("MSTrackPt", &MSTrackPt);
   tree->SetBranchAddress("MSTrackEta", &MSTrackEta);
   TH1F hmupt("hmupt", "Muon Pt", 100, 0, 500);
   for(Int_t i=0;i<tree->GetEntries();i++){
      tree->GetEntry(i);
      Int_t nmuon = muonPt->size();
      for (Int_t imuon = 0; imuon < nmuon; imuon++) { hmupt.Fill((*muonPt)[imuon]); }
   }
   hmupt.Draw();
}
```

そ 後 以下 コマ ド よってマ

```
$ root macro.C
```

同じこ 次 よ 一回root 立ち上&
.x
コマ ドで実行 こ もできま

```
$ root
root [] .x macro.C
```

こ 形式 マ 場合 今までClingで&
マ 内で定義し 変数(例えば&
だし 改行が文 終わり 認識

も 一つ 形式 コ ド 名前 つ &#38
例えば 次 よ なりま macro2.C
ファ ル 保存してくださ

入力が大変な人 こ ファ ル &

/home/sawada/public/tutorial16/macro2.C

```
Bool_t CheckDEta(Double_t eta1, Double_t eta2, Double_t threshold)
{
   return (fabs(eta1 - eta2) < threshold);
}

Int_t macro2(Double_t threshold = 0.01)
{
   TFile *file = TFile::Open("tree.root");
   TTree *tree = dynamic_cast<TTree*>(file->Get("tree"));
   Int_t EventNumber;
   Double_t METPt;
   Double_t METEta;
   vector<Double_t> *muonPt = 0;
   vector<Double_t> *muonEta = 0;
   vector<Double_t> *MSTrackPt = 0;
   vector<Double_t> *MSTrackEta = 0;
   tree->SetBranchAddress("EventNumber", &EventNumber);
   tree->SetBranchAddress("METPt", &METPt);
   tree->SetBranchAddress("METEta", &METEta);
   tree->SetBranchAddress("muonPt", &muonPt);
   tree->SetBranchAddress("muonEta", &muonEta);
   tree->SetBranchAddress("MSTrackPt", &MSTrackPt);
   tree->SetBranchAddress("MSTrackEta", &MSTrackEta);
   TH1F *hmupt = new TH1F("hmupt", "Muon Pt", 100, 0, 500);
   for(Int_t i=0;i<tree->GetEntries();i++){
      tree->GetEntry(i);
      Int_t nmuon    = muonPt->size();
      Int_t ntrack = MSTrackPt->size();
      for (Int_t imuon = 0; imuon < nmuon; imuon++) {
         for (Int_t itrack = 0; itrack < ntrack; itrack++) {
            if (CheckDEta((*muonEta)[imuon], (*MSTrackEta)[itrack], threshold)) {
               hmupt->Fill((*muonPt)[imuon]);
            }
         }
      }
   }
   hmupt->Draw();

   return 0;
}
```

ここで 一つ ファ ル 関数が&
実行時 関数 引数 渡 こ もで

```
$ root 'macro2.C(0.05)'
```

こ 場合 最初 紹介し 名前な&
(こ 例で hmupt newで作って 関数&#

ここで 説明 関数 でnewし ヒ&
(Clone 目的 関数で意図的 こ よ

ROOT マ コ パ ルしてから実行

*
完全 C++ コ パ ラ 準拠し 文法&#

*
Clingで 宣言されて な 変数が&#203

*
コ パ ル時 通常 コ パ ラ が(&#124

*
Debugオプ つけ 場合 ラッ ュ&#121

*
コ パ ラ 際 最適化 こ がで&#12

先ほど マ ACLiCで使え よ 次 &#124
違 ファ ル 先頭でヘッ ル
namespace
std; 入れ かど か 好み 問題だ &#24

```
#include <cmath>
#include <TFile.h>
#include <TTree.h>
#include <TH1.h>

using namespace std;

Bool_t CheckDEta(Double_t eta1, Double_t eta2, Double_t threshold)
{
   return (fabs(eta1 - eta2) < threshold);
}

Int_t macro2(Double_t threshold = 0.01)
{
   TFile *file = TFile::Open("tree.root");
   TTree *tree = dynamic_cast<TTree*>(file->Get("tree"));
   Int_t EventNumber;
   Double_t METPt;
   Double_t METEta;
   vector<Double_t> *muonPt = 0;
   vector<Double_t> *muonEta = 0;
   vector<Double_t> *MSTrackPt = 0;
   vector<Double_t> *MSTrackEta = 0;
   tree->SetBranchAddress("EventNumber", &EventNumber);
   tree->SetBranchAddress("METPt", &METPt);
   tree->SetBranchAddress("METEta", &METEta);
   tree->SetBranchAddress("muonPt", &muonPt);
   tree->SetBranchAddress("muonEta", &muonEta);
   tree->SetBranchAddress("MSTrackPt", &MSTrackPt);
   tree->SetBranchAddress("MSTrackEta", &MSTrackEta);
   TH1F hmupt("hmupt", "Muon Pt", 100, 0, 500);
   for(Int_t i=0;i<tree->GetEntries();i++){
      tree->GetEntry(i);
      Int_t nmuon    = muonPt->size();
      Int_t ntrack = MSTrackPt->size();
      for (Int_t imuon = 0; imuon < nmuon; imuon++) {
         for (Int_t itrack = 0; itrack < ntrack; itrack++) {
            if (CheckDEta((*muonEta)[imuon], (*MSTrackEta)[itrack], threshold)) {
               hmupt.Fill((*muonPt)[imuon]);
            }
         }
      }
   }

   return 0;
}
```

実行時 次 よ ファ ル名 後
+ 追加しま

```
$ root macro2.C+
```

さら そ 後 g
追加 バッグオプ 付き な&#1
o 追加 最適化されま

ス ドア プ グラム 書くよ&#12

C++プ グラムか&#1

ここで あまり深入りしま&#12
基本的 コ ド内で ROOT ラス &#20351
ROOT関連 初期化処理 自動的 &#348
コ パ ル リ フラグ `root-config --cflags`
`root-config --glibs` で得 こ ができま
独自 ラス ツリ 保存し 時 &#

複雑な関数で

実際 でピ フィッ 行 時 &#3202

定義済み関数 &#20351

そ 場合でもフィッ 手続き &
ブルガウス分布 次 よ 定義

```
root [] TF1 fdgaus("fdgaus", "gausn(0)+gausn(3)", -10, 10)
root [] fdgaus.SetParameters(100, 0, 1, 20, 0, 2)
root [] fdgaus.Draw()
```

TF1 コ ス ラ 二つ目 引数で関&#
SetParametersでフィッ 前 パラメ 初&
パラメ 意味 最初 三つが&#1996

```
root [] TH1F hist("hist", "Histogram;x;number of events", 100, -10, 10)
root [] for (int i = 0; i < 10000; i++) { hist.Fill(fdgaus.GetRandom()); }
root [] hist.Draw()
root [] hist.Fit("fdgaus")
```

よ フィッ こ ができま
(ここで ヒス グラム 関数 &#20351
例えば事前 分布 ミ が0 来 &#20

```
root [] fdgaus.FixParameter(1, 0)
root [] fdgaus.FixParameter(4, 0)
root [] hist.Fit("fdgaus")
```

しま FixParameter 最初 引数 パラメ

ま フィッ でパラメ が動け

```
root [] fdgaus.SetParLimits(2, 0, 3)
root [] fdgaus.SetParLimits(5, 0, 7)
root [] hist.Fit("fdgaus")
```

よ しま 引数 意味 パラメ

式 形 指定 場合

的な式 書 て関数 定義 こ &#12

```
root [] TF1 fdcos("fdcos", "[0]*exp(-x/[1])*cos(x)", 0, 100)
root [] fdcos.SetParameters(20, 10)
root [] fdcos.Draw()
```

```
[0] [1]
```
が関数 パラメ 表して て &#1241
```
[1]
```
までな で 自動的 二つ パラ&

C++ 関数 使 ばあ

さら 複雑 なり 1行で書く が
func.C
ファ ル 開 て以下 内容 書 &#1

/home/sawada/public/tutorial16/func.C

```
Double_t ExpGaus(Double_t *x, Double_t *par)
{
   //http://pibeta.phys.virginia.edu/~pibeta/docs/publications/penny_diss/node35.html
   //par[0] : height of Gaussian
   //par[1] : peak position
   //par[2] : sigma of Gaussian
   //par[3] : transition point between gaussian and exponential

   Double_t fitval;

   if(x[0] > par[1] + par[3]) {
      if(par[2] != 0) {
         fitval = par[0] * TMath::Exp(-1 * (x[0] - par[1]) * (x[0] - par[1]) / 2 / par[2] / par[2
      } else {
         fitval = 0;
      }
   } else {
      if(par[2] != 0) {
         fitval = par[0] * TMath::Exp(par[3] / par[2] / par[2] * (par[3] / 2 - (x[0] - par[1])));
      } else {
         fitval = 0;
      }
   }

   return fitval;
}
```

これ あ x 境 ガウス分布 指&#259
パラメ ４つあり 関数 先頭
それで こ C++ 関数 使って ROOT &#38

```
root [] .L func.C
root [] TF1 fegaus("fegaus", ExpGaus, 0, 100, 4)
root [] fegaus.SetParameters(100, 50, 2, -1)
root [] fegaus.Draw()
```

1行目で ExpGaus
関数 読み込んで ま こ 形式
こ よ TF1 使 場合 C++ 関数 引数 &#36

さら 難し Fit

今まで紹介し 方法で 1次元 &
これら つ て こ 講習 範疇 &#362

発展

こ 講習で ROOT ほん 一部だけ&#3
実際 Graphical User
interface ソフ ウェア開発 ステ&#1251

練習

練習 して 実際 ATLAS で使われ

/home/sawada/public/tutorial16/sample.root Zがee 崩壊 MCから作
元々 目的 Tag-and-probe法 よって 電&#23
Zee ち 一方 再構成され (陽)電&#2
Probe側 (陽)電子 必ずしも ラッ &#

練習 して こ ファ ル 使って

* Tag側 Probe側 ファ 方向 散布図

*
電子 陽電子で組んだ不変&#36

作ってくださ

以下 ブラ チ 使ってくださ

グ側Pt : ZeeProbClusterTagPt

グ側Eta : ZeeProbClusterTagEta

グ側Phi : ZeeProbClusterTagPhi

プ ブ側Pt : ZeeProbClusterProbeClusterPt

プ ブ側Eta : ZeeProbClusterProbeClusterEta

プ ブ側Phi : ZeeProbClusterProbeClusterPhi

不変質量 答え: ZeeProbClusterM

これら ブラ チ 型 vector<float>で
で Zeeらし Tag Probe ペア なって で

不変質量 計算 自分で書 て&#
TLorentzVector等 便利な ラス 使って&#124
これら 使 場合 ROOT Class
reference 参照してくださ

https://root.cern.ch/doc/master/classTVector3.html

https://root.cern.ch/doc/master/classTLorentzVector.html

This topic: Main > AtlasJapanSoftwareTutorial16DecROOT
Topic revision: r11 - 2016-12-28 - RyuSawada