高エネルギー宇宙物理学のための ROOT 入門

– 第 2 回 –

奥村 曉

名古屋大学 宇宙地球環境研究所

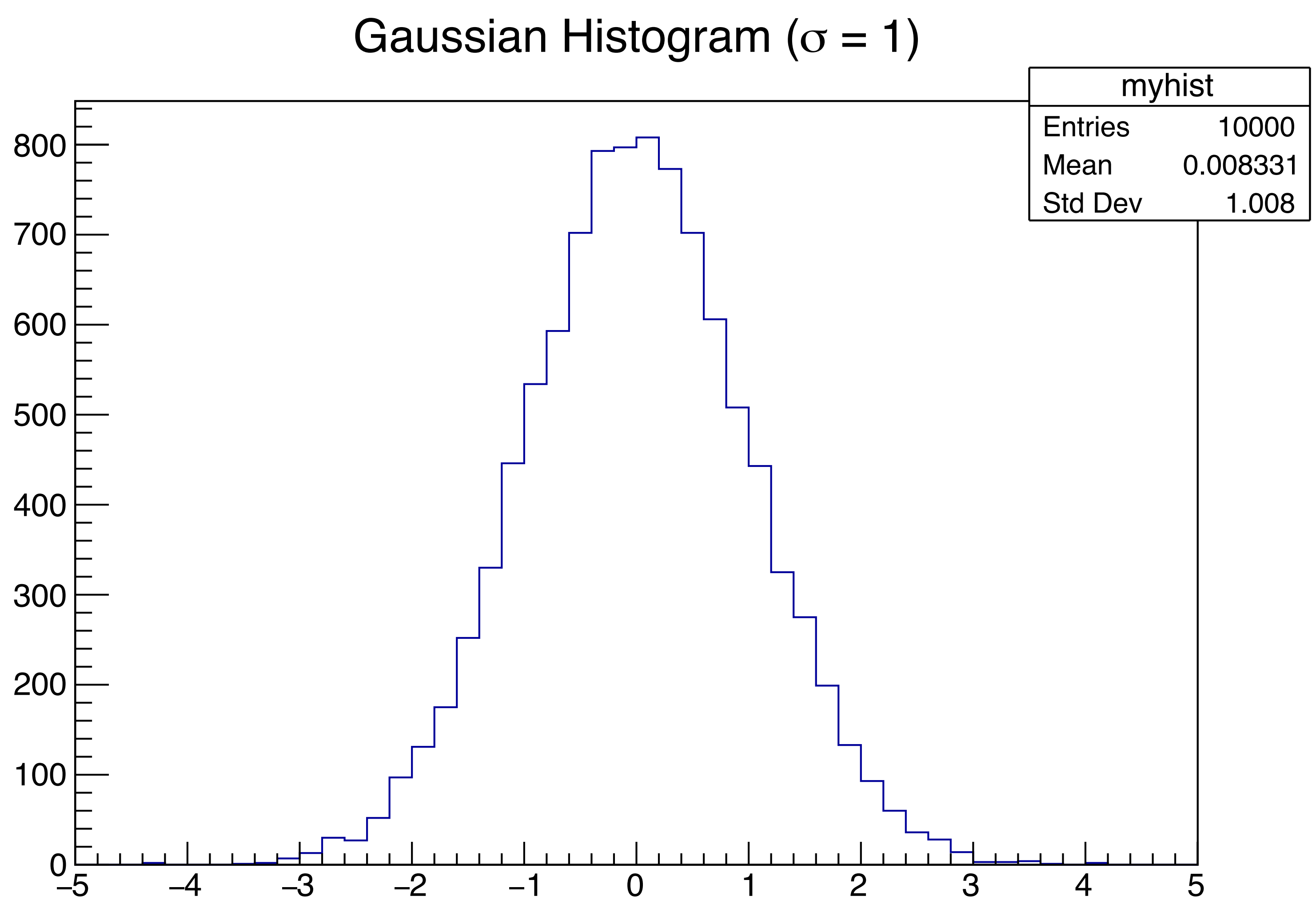
2019 年 4 月 24 日

名古屋大学 NAGOYA UNIVERSITY

ISEE Institute for Space–Earth Environmental Research

oxon@mac.com

復習：正規分布のヒストグラム




```
$ root
root [0] TH1D* hist = new TH1D("myhist", "Gaussian Histogram (#sigma = 1)", 50, -5, 5)
root [1] hist->FillRandom("gaus", 10000)
root [2] hist->Draw()
Info in <TCanvas::MakeDefCanvas>:  created default TCanvas with name c1
root [3] c1->SaveAs("hist.pdf)
```

❶ 目的に応じてグラフやヒストグラムを作る

❷ データ解析の操作をする

❸ 描画する

❹ 図に保存する

✓ROOT インタプリタは C++ を解釈できる

✓ROOT プロンプトに 1 行ずつ入力し実行する

復習：少し解説

```
root [0] TH1D* hist = new TH1D("myhist", "Gaussian Histogram (#sigma = 1)", 50,
-5, 5)
```

❶ TH1D という 1 次元ヒストグラム用クラスのインスタンス（オブジェクト）を新たに作る
❷ コンストラクタの引数で、その仕様を決める

```
root [1] hist->FillRandom("gaus", 10000)
```

❸ オブジェクトに対して色々と操作をする。この場合は、ガウス分布（正規分布）で乱数を 10,000 回詰める

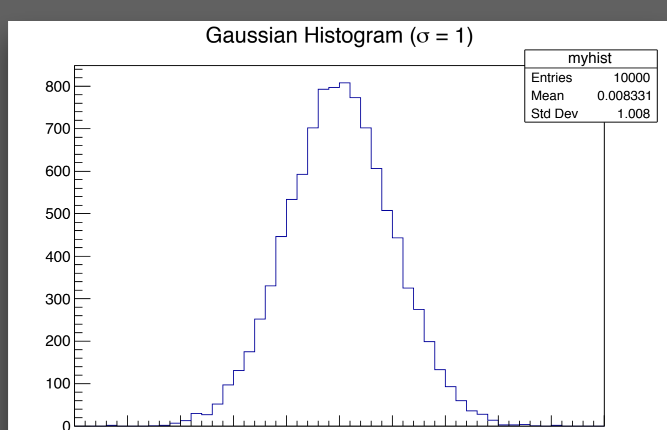
```
root [2] hist->Draw()
```

❹ 多くの ROOT クラスは、Draw() というメソッド（メンバ関数）を呼んで描画することができる

```
Info in <TCanvas::MakeDefCanvas>:  created default TCanvas with name c1
```

❺ 描画先（キャンバス）を作っていないので、デフォルトの大きさのものが c1 という名前で自動的に作られる

```
root [3] c1->SaveAs("hist.pdf")
```



❻ Draw した図は、色々な画像形式で保存可能。c1 も ROOT のオブジェクトなので、メソッドを多く持つ。
基本的に PDF で保存すること（EPS は今どき使わない）。

new とポインタ

❶ new を使うやり方

```
$ root
root [0] TH1D* hist = new TH1D("myhist", "Gaussian Histogram (#sigma = 1)", 50,
-5, 5)
```

もしくは（書き方が違うだけで同じこと）

```
root [0] TH1D* hist
root [1] hist = new TH1D("myhist", "Gaussian Histogram (#sigma = 1)", 50, -5, 5)

root [1] hist->FillRandom("gaus", 10000)
root [2] hist->Draw()
Info in <TCanvas::MakeDefCanvas>:  created default TCanvas with name c1
root [3] c1->SaveAs("hist.pdf")
```

❷ new を使わないやり方

```
$ root
root [0] TH1D hist("myhist", "Gaussian Histogram (#sigma = 1)", 50, -5, 5)
root [1] hist.FillRandom("gaus", 10000)
root [2] hist.Draw()
Info in <TCanvas::MakeDefCanvas>:  created default TCanvas with name c1
root [3] c1->SaveAs("hist.pdf")
```

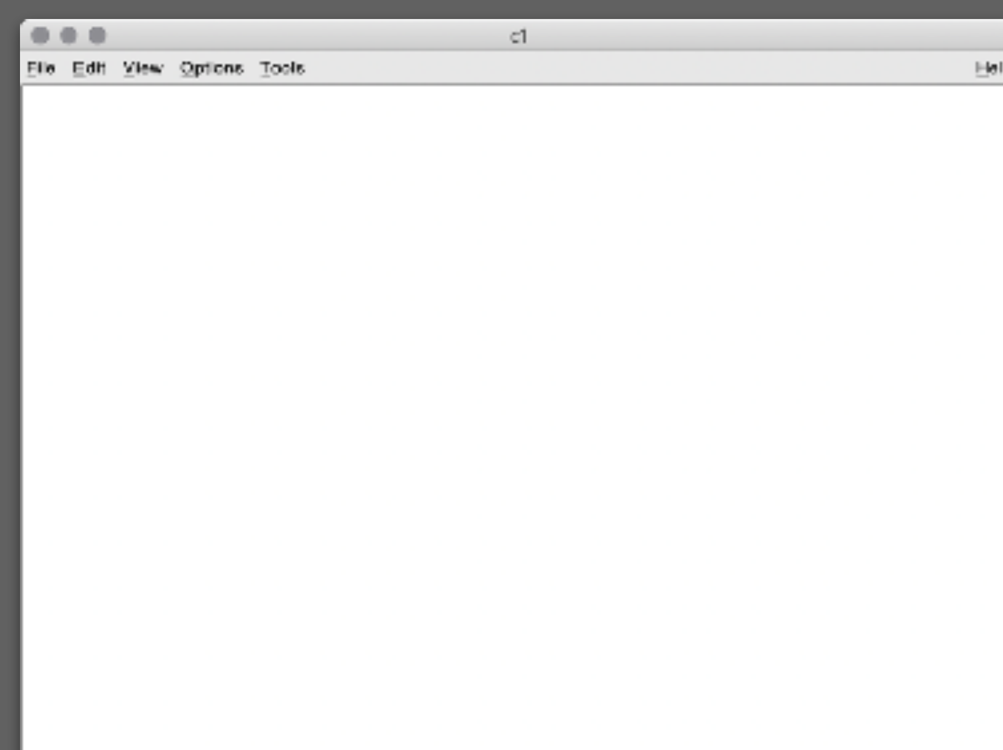
{} で囲むと自動で消えてしまう

```
$ root
root [0] {
root (cont'ed, cancel with .@) [1] TH1D hist("myhist", "Gaussian Histogram
(#sigma = 1)", 50, -5, 5);
root (cont'ed, cancel with .@) [2]}
root [3] hist
input_line_21:2:3: error: use of undeclared identifier 'hist'
 (hist)
  ^
Error in <HandleInterpreterException>: Error evaluating expression (hist).
Execution of your code was aborted.
```

new を使わないとどうなるか

```
$ cat first_script4.C

void first_script4() {
  TH1D hist("myhist", "Gaussian Histogram (#sigma = 1)", 50, -5, 5);
  hist.FillRandom("gaus", 10000);
  hist.Draw();
}


$ root
root [0] .x first_script4.C
```

ヒストグラムが {} を抜けると消えてしまう



new したら本当は delete しないといけない

```
root [0] TH1D* hist = new TH1D("myhist", "Gaussian Histogram (#sigma = 1)", 50,
-5, 5)
root [1] hist->FillRandom("gaus", 10000)
root [2] hist->Draw()
Info in <TCanvas::MakeDefCanvas>:  created default TCanvas with name c1
root [3] c1->SaveAs("hist.pdf")

root [4] delete hist

root [5] hist = 0

root [6] .q
```

❶ delete でメモリ上から hist の実体を消す

❷ hist には実体がないことをプログラム内の他の場所に伝えるため、ゼロを詰める

ざっくりとした new のまとめ

- C++ の初学者は new を使う場面は多くなく、教科書でも後ろのほうで出てくる
- 非常に簡単に説明すると
  - new を使う場合はメモリ管理を自分でする必要がある
  - 他の場合は、計算機がメモリ管理をしてくれる
- ROOT で解析する場合は、勝手にオブジェクトが消えると困る場合が多いので、new を使うことが多い
- 本当は delete をしないといけない
- この辺り、ROOT のみで C++ を学ぶと C++ プログラムに大量のバグを生み出すので、別途 C++ の教科書で学習してください

ROOT ファイルとして保存する

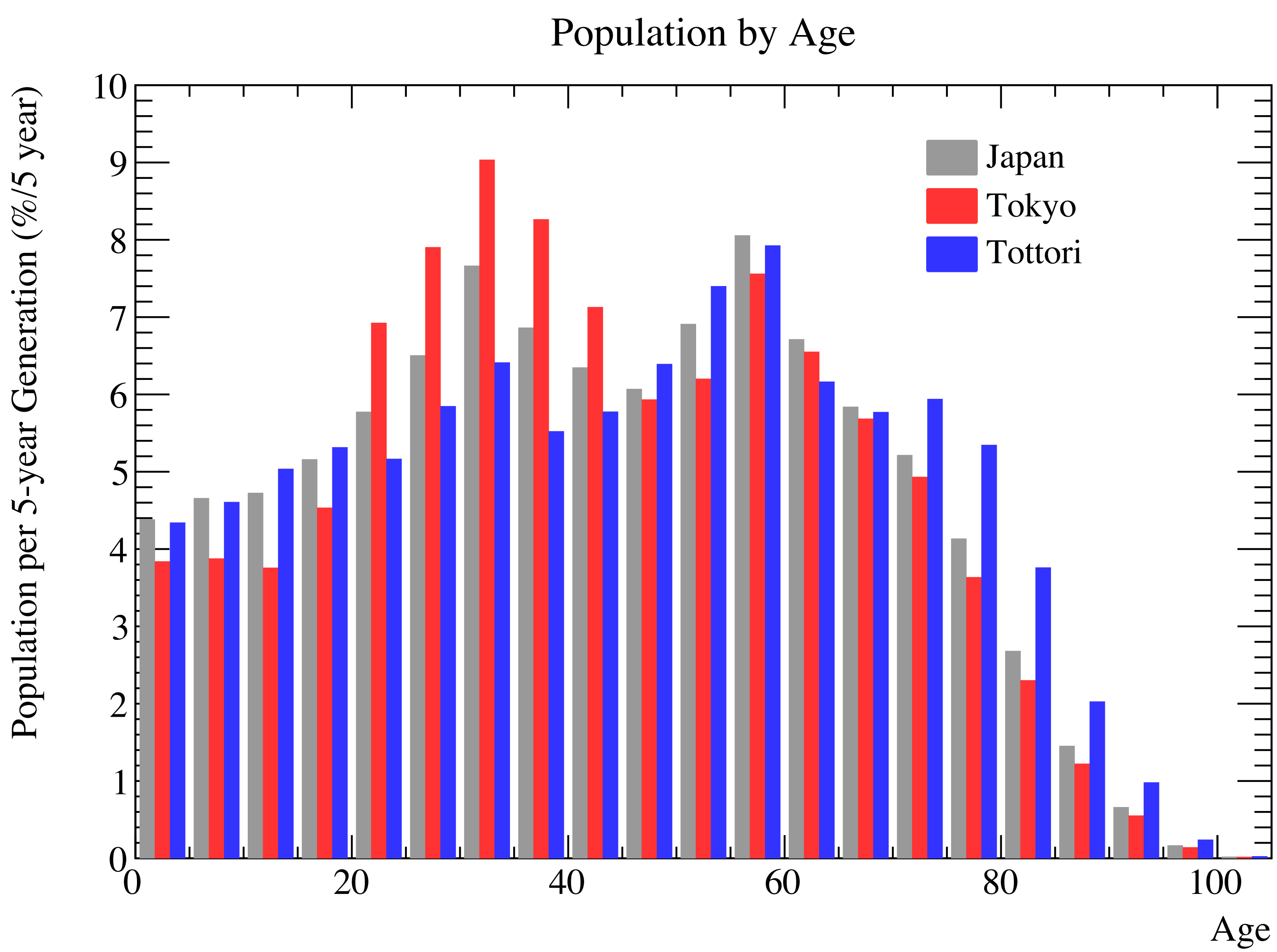
```
root [4] c1->SaveAs("hist.root")
root [5] .q

$ root
root [0] TFile f("hist.root")
(TFile &) Name: hist.root Title:

root [1] f.ls()
TFile**      hist.root
 TFile*      hist.root
  KEY: TCanvas   c1;1c1

root [2] c1->Draw()

root [3] TH1D* hist = (TH1D*)c1->GetPrimitive("myhist")
```

1. SaveAs で ROOT ファイルとして保存する
2. TFile を使って ROOT ファイルを開く
3. 先ほど作った c1 というキャンバスが存在
4. 再度 Draw できる
5. 再度 hist を触れる

- 解析結果は必ず描画して問題ないか自分の目で確認する
- 論文に使ったり人に渡したり印刷しやすい PDF に保存する
- 後から修正したりヒストグラム自体の操作などができるよう ROOT ファイルにも保存する

ヒストグラム

ヒストグラム（histogram）とはなにか？



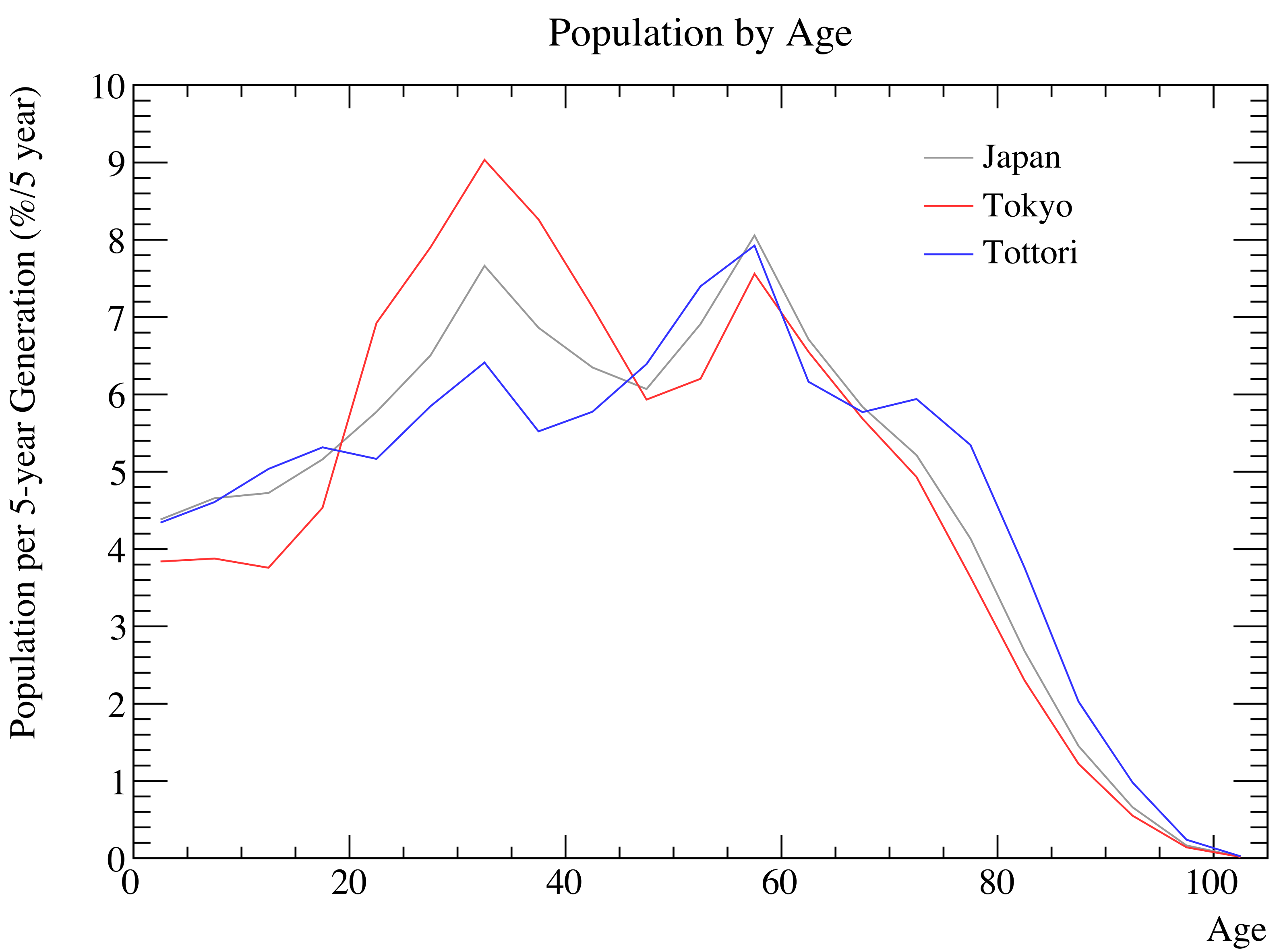
全数調査のヒストグラムの例（ビン幅は 5 歳）
（データ出典：国勢調査 2005）

- 度数分布図
- ある物理量がどのように分布しているかを、値の範囲をビン（bin）に区切って表示したもの
- 実験での使用例
  - 光検出器の波高分布（ポアソン分布と正規分布）
  - 崩壊時間や飛程の分布（指数分布）
- 分布同士の比較、理論曲線との比較によく使われる

大事なこと

- 積分すると総数になる
  - 縦軸は明示的に書かれていなくとも、per bin である
  - 標本の大きさ（sample size）
    - 総測定回数や総発生事象（トリガーした宇宙線粒子のエネルギー分布など）
    - 全数測定の総数（国勢調査、実験装置の全数調査など）
    - 標本数（number of samples）とは言わないので注意
  - 確率密度関数の場合は 100% や 1
    - 十分に標本が大きい（=統計誤差の小さい）MC シミュレーションで得られた物理量の分布や理論曲線など
- 面積に意味があるので原則として縦軸のゼロを表示する（対数表示の場合はもちろん不可能）
- 全数調査と標本調査は分布が異なる

間違った表示の例



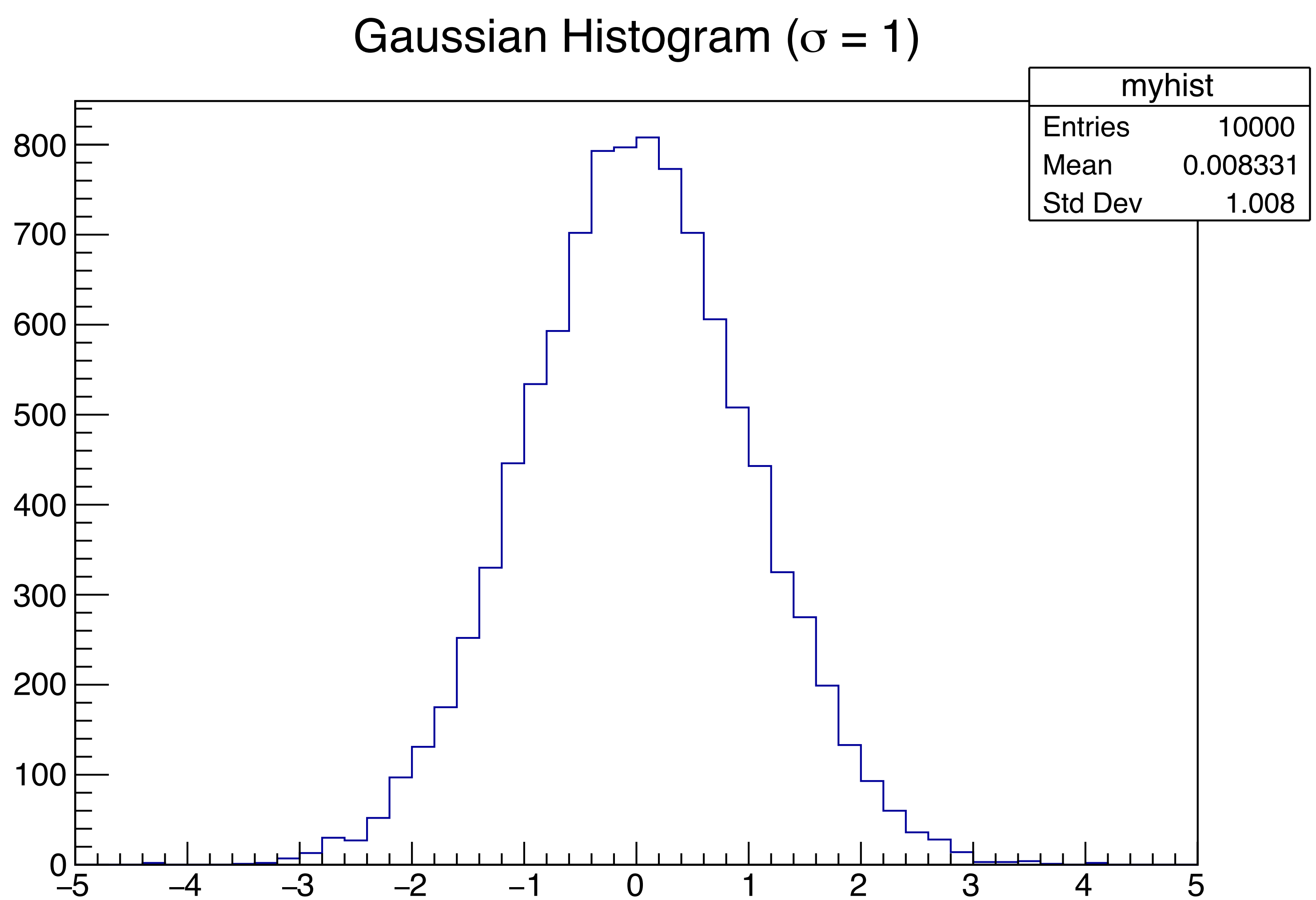
- ヒストグラムを折れ線グラフにしない
  - ビンの中心値はそのビンの代表値ではない
  - 面積が保存しない
  - （多くの場合）折れ線の傾きに物理的な意味がない
  - 誤差棒が大きい場合、傾きを見せるのは読者の誤解を誘発する

1 次元ヒストグラム

TH1 クラス

- ROOT の 1 次元ヒストグラムは TH1 というクラス
- ヒストグラムの縦軸のデータ型に応じて複数の派生クラスがある
  - TH1D – double（14 桁まで扱える、多分一番よく使う）
  - TH1F – float（7 桁）
  - TH1C – char（-128〜+127）
  - TH1S - 16 bit int（short）（-32768〜+32767）
  - TH1I – 32 bit int（-2147483648〜+2147483647）
- TH1D 以外はひとまず忘れて良い

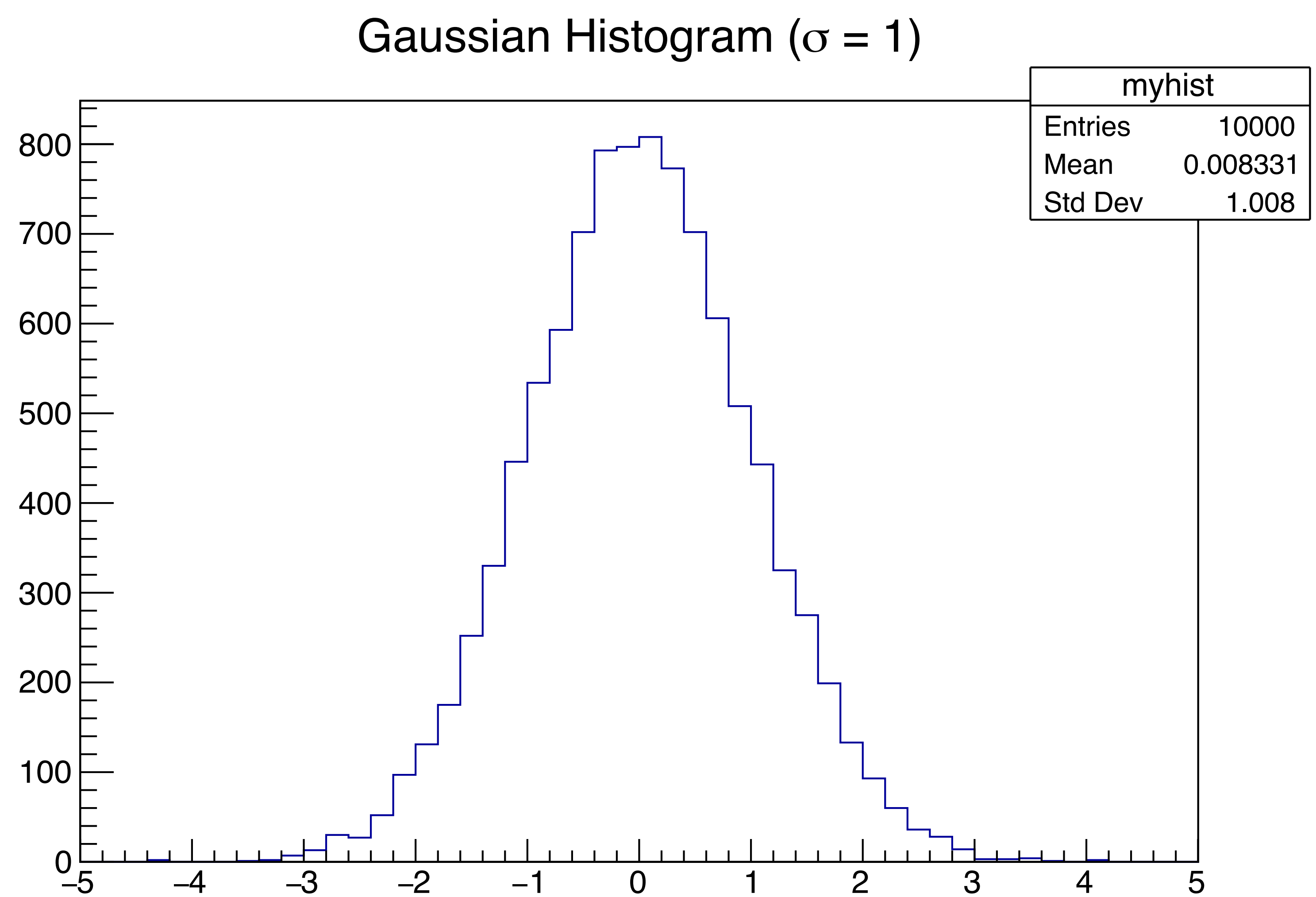
少し前に戻る




```
$ root
root [0] TH1D* hist
           = new TH1D("myhist",
                      "Gaussian Histogram (#sigma = 1)",
                      50,
                      -5,
                      5)
root [1] hist->FillRandom("gaus", 10000)
root [2] hist->Draw()
```

❶ 名前 (なくてもよい)
❷ タイトル (なくてもよい)
❸ ビン数
❹ 下限値
❺ 上限値
❻ 標準偏差 1 の正規分布を乱数で 10^4 回詰める

自分で 10^4 回詰める

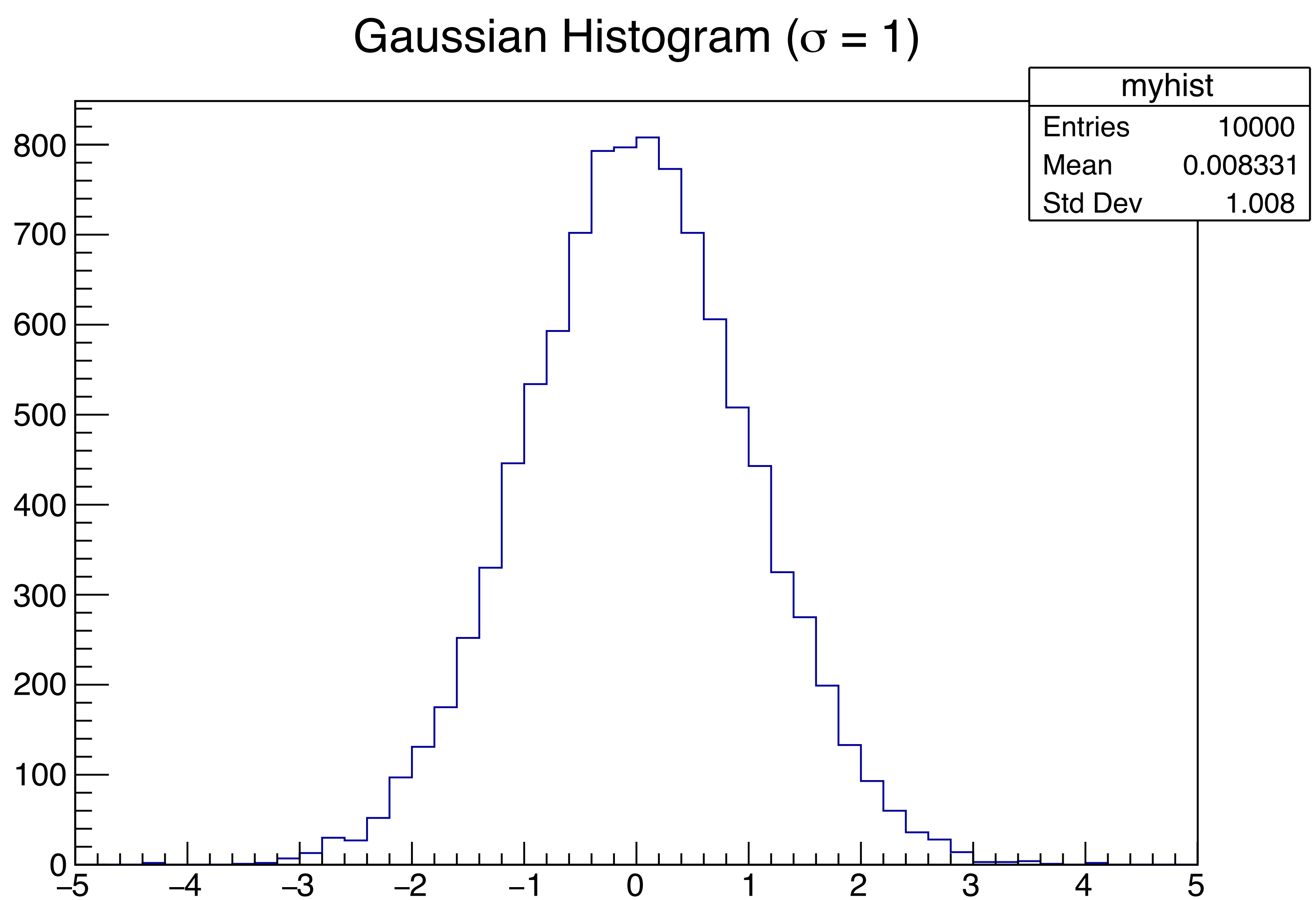



```
$ root
root [0] TH1D* hist = new TH1D("myhist", "Gaussian Histogram (#sigma = 1)", 50,
-5, 5)
root [1] for(Int_t i = 0; i < 10000; i++){
root (cont'ed, cancel with .@) [2] Double_t x = gRandom->Gaus();
root (cont'ed, cancel with .@) [3] hist->Fill(x);
root (cont'ed, cancel with .@) [4]}
root [5] hist->Draw()
```

❶ 乱数を生成し
❷ 詰める

※ 実際には、測定値などを詰める

ヒストグラムの基本的な量




```
root [6] hist->GetEntries()
(Double_t) 10000.0
root [7] hist->GetMean()
(Double_t) 0.008331
root [8] hist->GetStdDev()
(Double_t) 1.008
root [9] hist->GetMeanError()
(Double_t) 0.00997350
root [10] hist->GetStdDevError()
(Double_t) 0.00705233
```

❶ 総数

❷ 標本の平均値（母集団の真の平均値ではない）

❸ 標本の標準偏差（standard deviation）

❹ 平均値の統計誤差

❺ 標準偏差の統計誤差

平均値、分散、標準偏差

- 平均値：通常、ある物理量の相加平均（母平均は μ）

標本平均 $$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i = \frac{x_1 + x_2 + \cdots x_N}{N}$$

母平均 $$\mu = \lim_{N\to\infty}\frac{1}{N}\sum_{i=1}^{N} x_i$$

- 分散：その分布の散らばり具合を示す

（不偏）標本分散 $$s^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2$$ ※ ROOT は N で割っている

母分散 $$\sigma^2 = \lim_{N\to\infty}\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2$$

- 標準偏差：散らばり具合を物理量と同じ次元で示す

標本の標準偏差 s　　母集団の標準偏差 σ

RMS と用語を混同しないこと

- RMS（二乗平均平方根）と標準偏差は定義が異なります

$$\mathrm{RMS} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2}$$

※ 平均を引かない

- PAW や ROOT ユーザの多くが混同しているので注意
  - PAW が最初に間違い、ROOT は意図的に間違いを継承した
  - 最新の ROOT では、RMS という言葉はもう使われない
  - PD の 1 年目まで自分も勘違いしていた

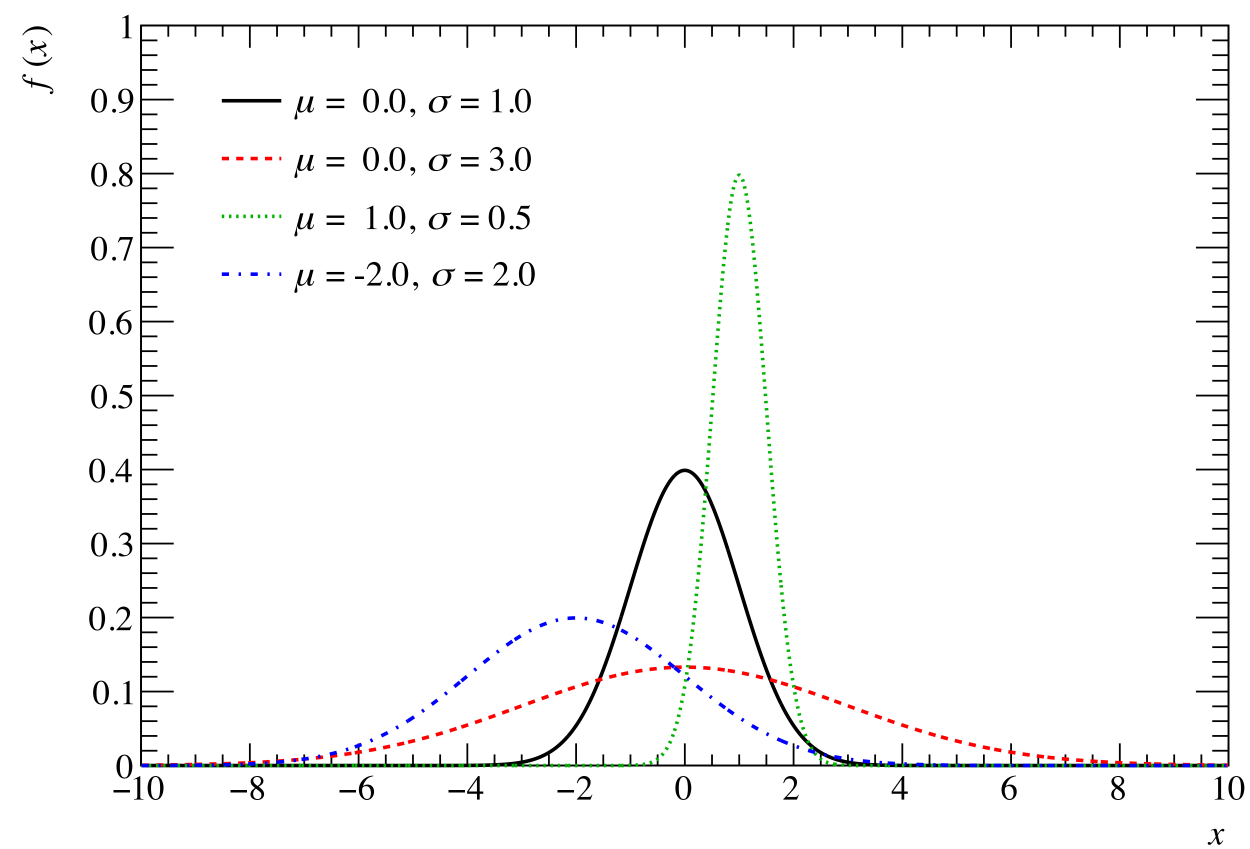
正規分布

正規分布（Normal Distribution）とは

- ガウス分布（Gaussian distribution）とも
- 平均値 μ と分散 σ^2（もしくは標準偏差 σ）で表される

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

- 我々が最も頻繁に使う分布
- 多数の確率過程が組み合わさった場合、結果として出てくる物理量が正規分布に従う（中心極限定理）
- 面積一定の場合、高さと幅は $1/\sigma$ と σ に比例する



GetMeanError と GetStdDevError

- 母集団の分布や理論的な分布が正規分布であったとしても、限られた実験データ（標本）は母集団を完全に再現しない
- 標本から得られる平均値や標準偏差は、真の値とはずれる
- TH1::GetMeanError と GetStdDevError は、そのずれの推定量を返す
- 正規分布の場合、物理量 x に対し次の推定量の誤差があることが知られている

$$\delta\bar{x} = \frac{s}{\sqrt{N}} \qquad \delta s = \frac{s}{\sqrt{2N}}$$

確かめてみる

```
$ cat StandardError.C
void StandardError() {
  const Int_t kSampleSize = 10000;
  const Int_t kRepeat = 10000;
  const Double_t kMean = 0.;
  const Double_t kSigma = 1.;

  TH1D* hMeanError = new TH1D("hMeanError", ";<#it{x}>", 100, -0.05, 0.05);
  TH1D* hStdDevError = new TH1D("hStdDevError", ";#it{#sigma}_{#it{x}}", 100, -0.05, 0.05);

  for(Int_t i = 0; i < kRepeat; i++){
    TH1D h("", "", 100, -5, 5);
    for(Int_t j = 0; j < kSampleSize; j++){
      Double_t x = gRandom->Gaus(kMean, kSigma);
      h.Fill(x);
    }
    hMeanError->Fill(h.GetMean() - kMean);
    hStdDevError->Fill(h.GetStdDev() - kSigma);
  }

  TCanvas* can = new TCanvas("can", "can", 1200, 600);
  can->Divide(2, 1, 1e-10, 1e-10);
  can->cd(1);
  hMeanError->Draw();
  can->cd(2);
  hStdDevError->Draw();
}
```

❶ 平均 $\mu = 0$、標準偏差 $\sigma = 1$

❷ 真の値からどれだけずれたかを詰めるヒストグラム

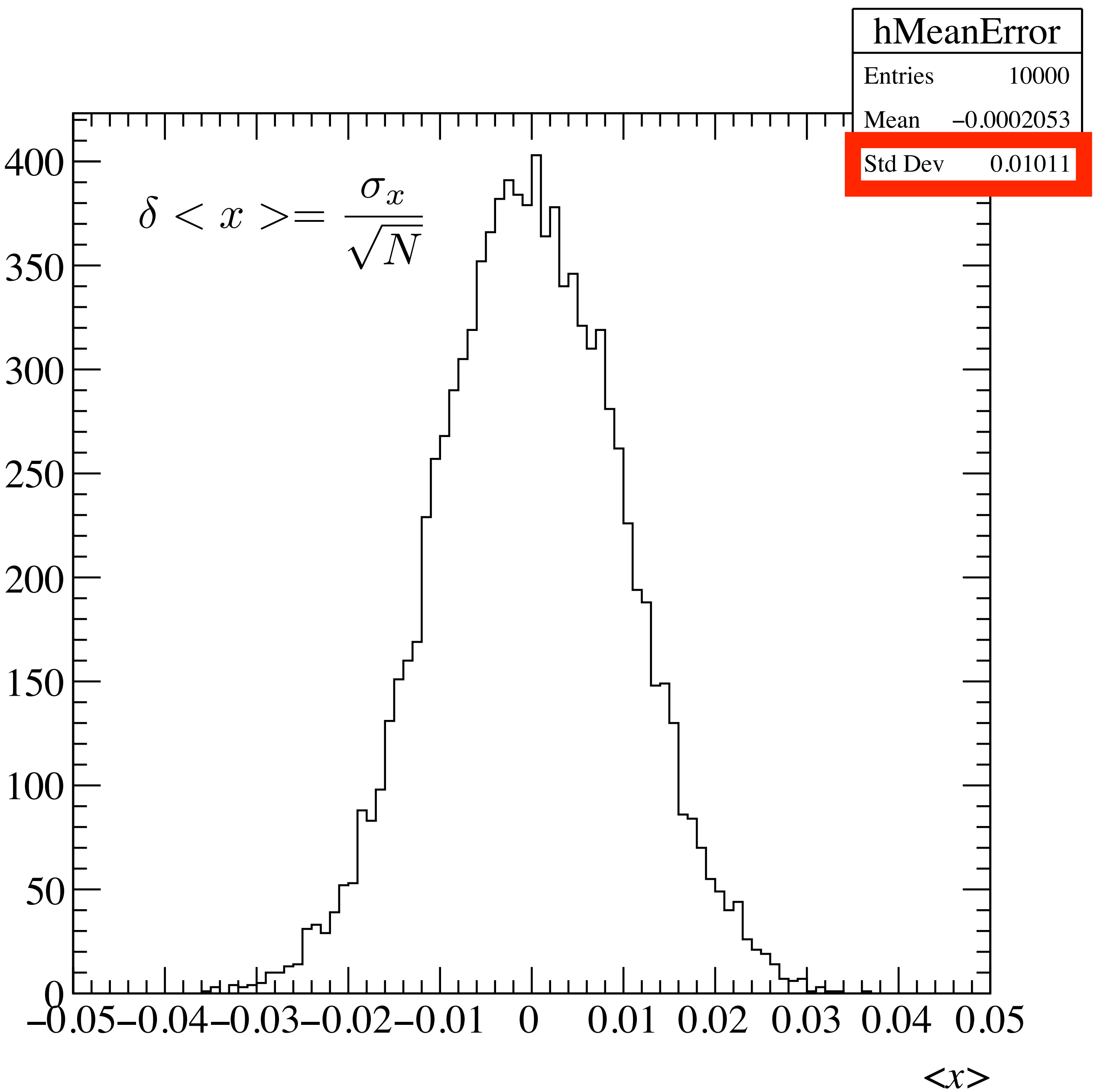
❸ $\mu = 0$、$\sigma = 1$ で乱数を 10,000 回生成

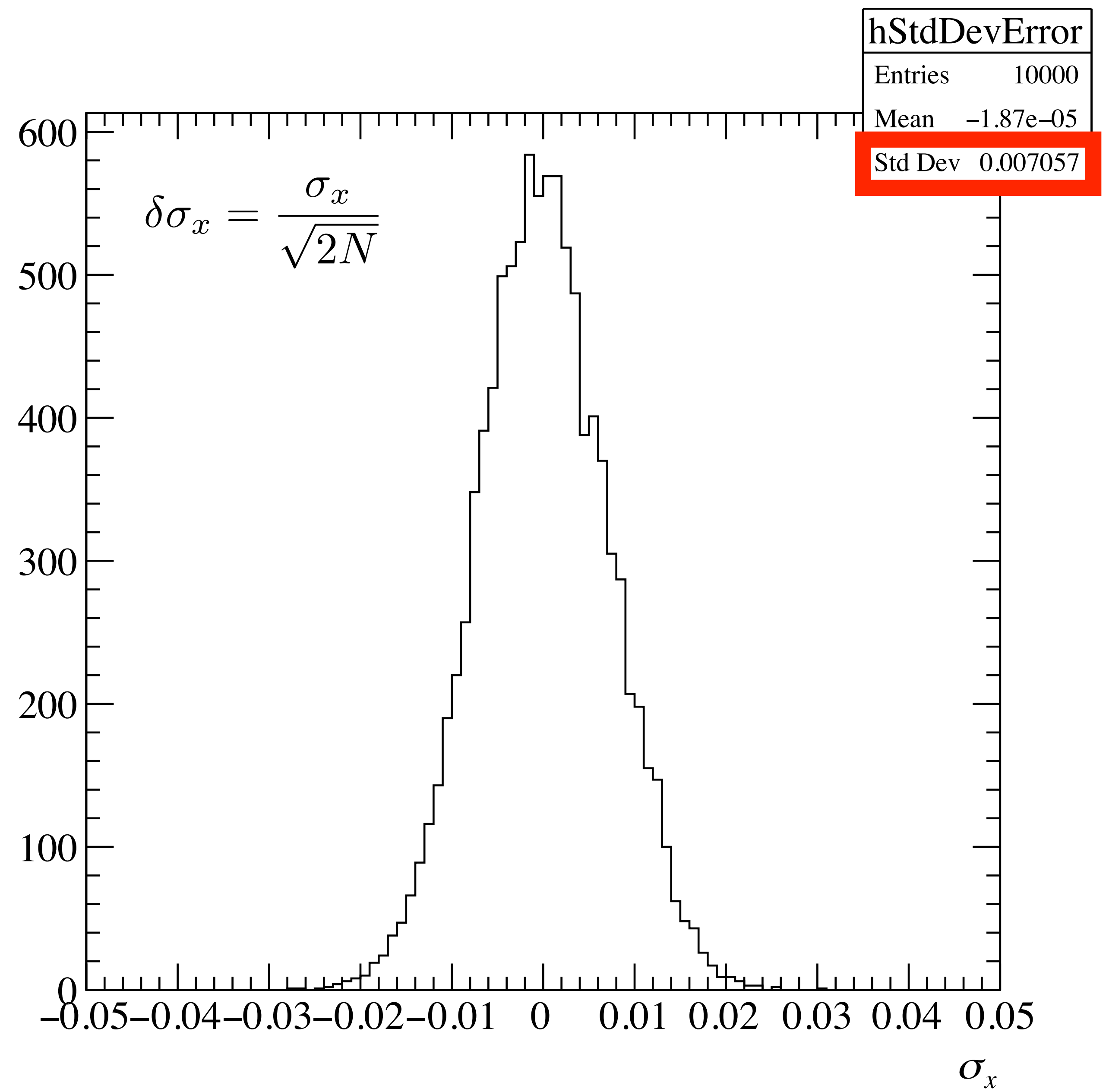
❹ 標本で得られた $\bar{x}$ と σ_x の、真値との差を詰める

これも 10,000 回繰り返し

❺ Draw する

確かめてみる





- $\sigma x/\sqrt{N} = 1/100 = 0.01$
- $\sigma x/\sqrt{2N} = 1/(1.4\cdots \times 100) = 0.0707$
- 誤差の範囲で一致している

大事なこと

- 通常の測定は母集団から標本を抜き出しているだけ
- 真の分布は知りえないので標本から推定する
- 平均値や標準偏差は、標本から計算されたもの
  - 真の平均値からの誤差は $\sigma_x/\sqrt{N}$
  - 真の標準偏差からの誤差は $\sigma_x/\sqrt{2N}$

- ある確率分布に従う測定があった場合、統計誤差はその分布の標準偏差
- 多数の測定から平均値を求める場合は、統計誤差は $\sigma_x/\sqrt{N}$

注意事項

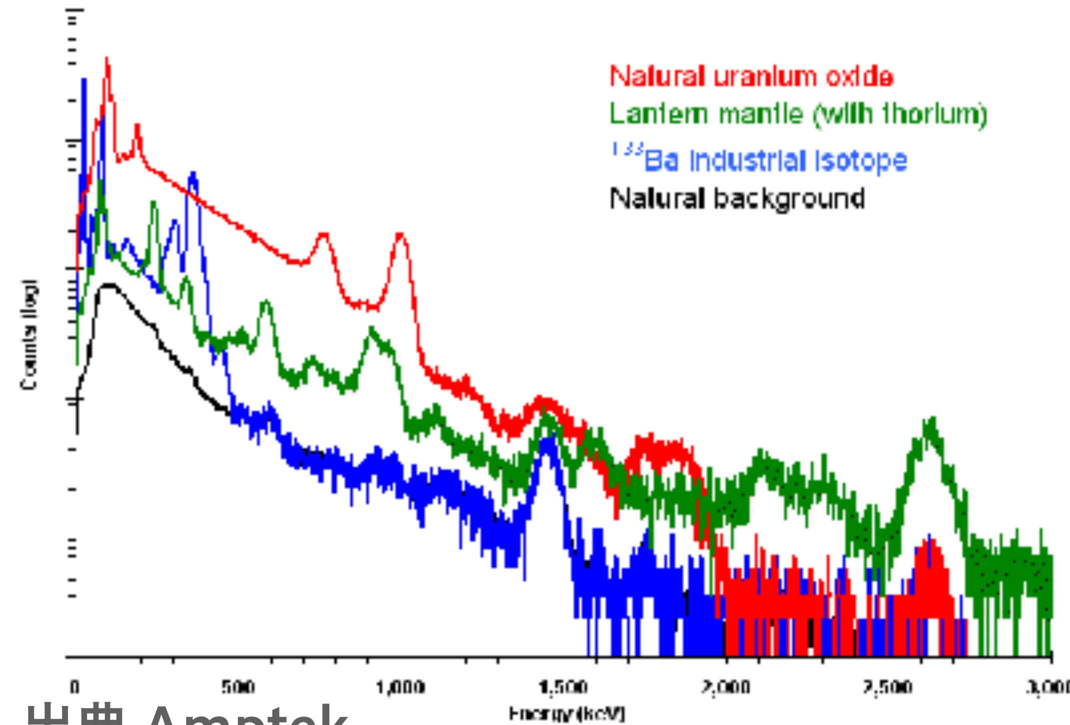
- 実際に実験データを解析する場合、真に正規分布であることはほとんどない
  - 正規分布は正負の無限大の値を取りうるが、実際の測定でそのような値は取りえない
  - 光電子増倍管の出力波高を正規分布と仮定することがあるが、負のゲインはありえない。また正規分布と異なることも実はよく知られている。

- ROOT で横軸の表示範囲を変更すると、平均値や標準偏差が表示範囲のみで再計算される
- ポアソン分布や指数分布などもあるので、各自勉強してください

フィッティング

ヒストグラムのフィット

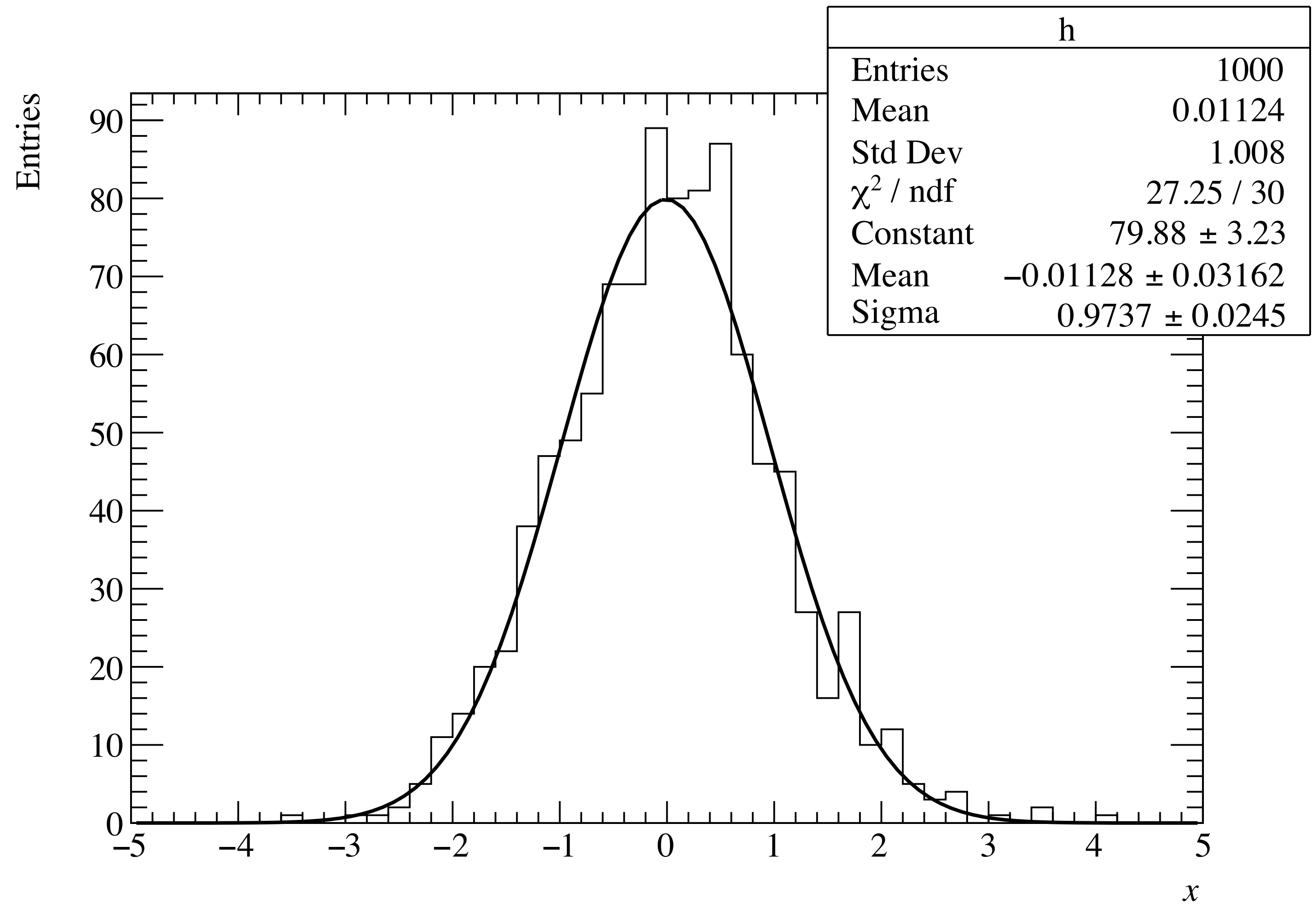
- 実験で得られたヒストグラムから物理量を抜き出すとき、単純な１つの正規分布であることは少ない
  - 複数のピークの存在するデータ
  - バックグラウンドを含むデータ



出典 Amptek
http://amptek.com/products/gamma-rad5-
ray-detection-system/

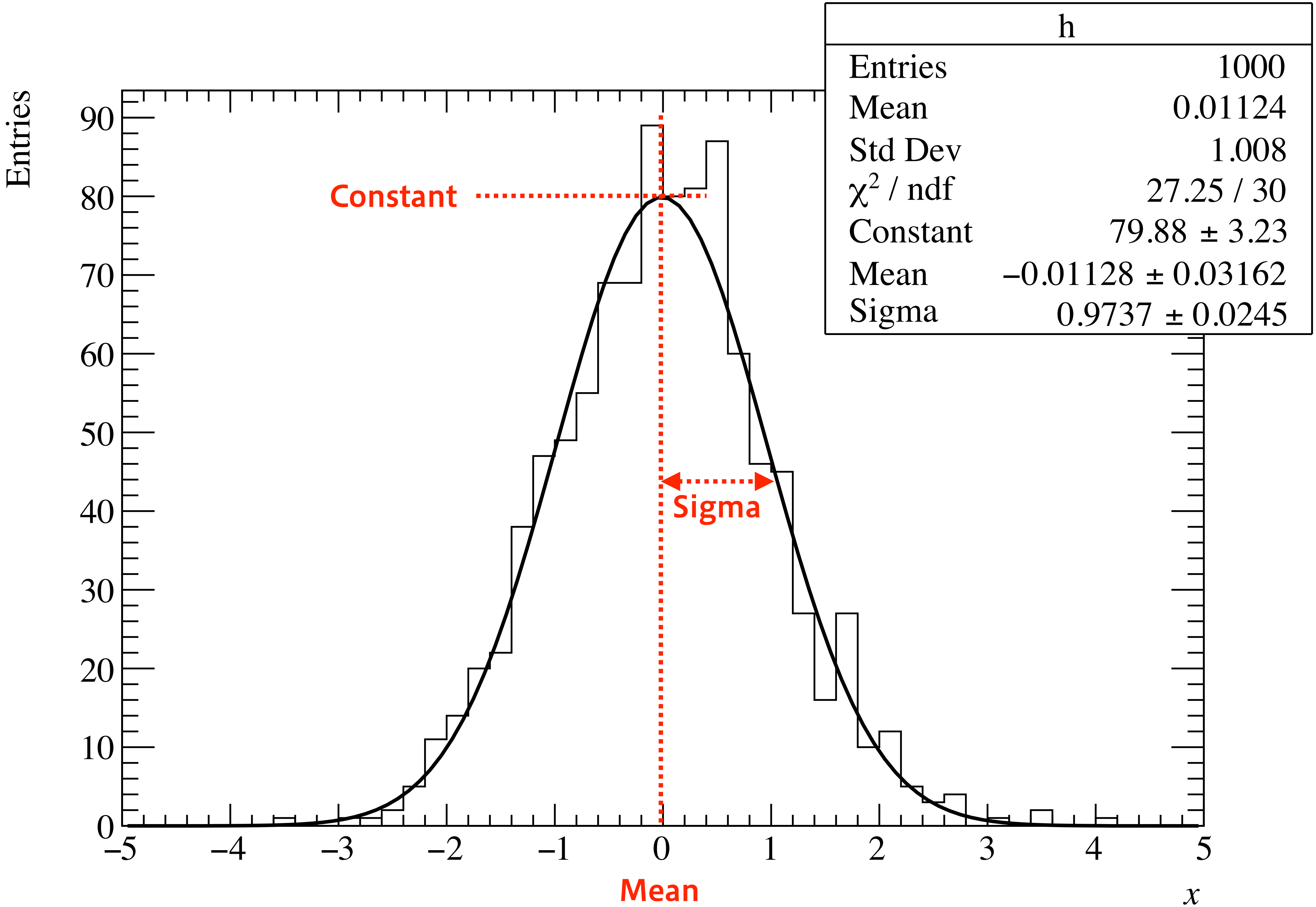
- ヒストグラムをよく再現するモデル関数を作り、フィット（fit、曲線のあてはめ）を行うことで変数（parameter）を得る

単純な例




```
root [0] TH1D hist("h", ";#it{x};Entries", 50, -5, 5)
root [1] hist.FillRandom("gaus", 1000)
root [2] hist.Fit("gaus")
 FCN=27.2533 FROM MIGRAD    STATUS=CONVERGED      60 CALLS           61 TOTAL
                     EDM=1.22437e-07    STRATEGY= 1      ERROR MATRIX ACCURATE
  EXT PARAMETER                                   STEP         FIRST
  NO.   NAME      VALUE            ERROR          SIZE      DERIVATIVE
   1  Constant     7.98846e+01   3.22837e+00   6.64782e-03  -1.29981e-05
   2  Mean        -1.12836e-02   3.16206e-02   8.19052e-05  -1.55071e-02
   3  Sigma        9.73719e-01   2.44588e-02   1.69219e-05  -7.15963e-03
```

詳細



h
Entries 1000
Mean 0.01124
Std Dev 1.008
χ² / ndf 27.25 / 30
Constant 79.88 ± 3.23
Mean −0.01128 ± 0.03162
Sigma 0.9737 ± 0.0245
Entries
Constant
Sigma
Mean
x

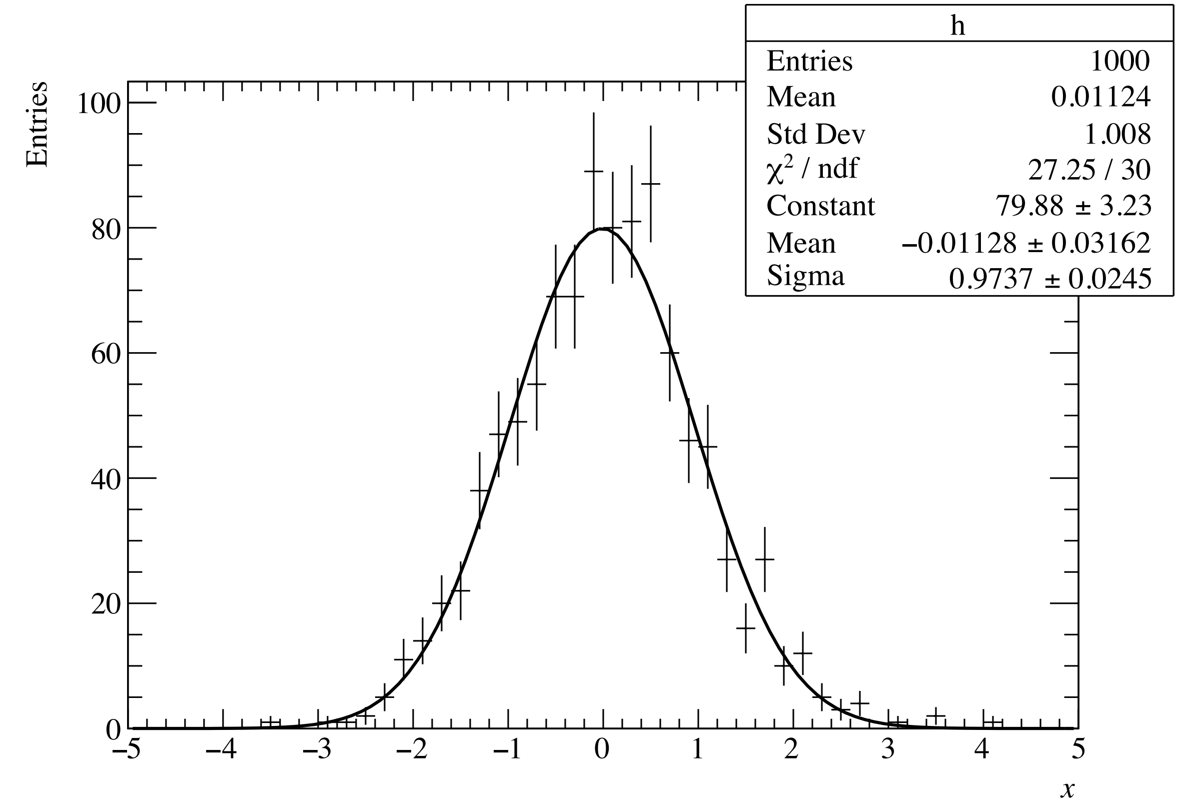
変数の比較

	平均	標準偏差
真値	0	1
ヒストグラム	0.011±0.032	1.008±0.023
フィット	-0.011±0.032	0.974±0.025

- 両者とも誤差の範囲で真値を推定できている
- 誤差の大きさは両者で同程度

ROOT は内部で何をしているか

- 各ビンには統計誤差が存在
  - そのビンに入る標本の大きさはポアソン分布に従う
  - N > 20 で正規分布と見なせる
  - $\delta N = \sqrt{N}$ と近似できる



- 最小二乗法を用いて、カイ二乗（χ^2）を最小にするように、モデル関数の変数空間を探索する

$$\chi^2 = \sum_{i=1}^{N} \frac{(y_i - f(x_i))^2}{\delta y_i^2}$$

x_i : ビンの中心値
y_i : 各ビンの計数
$f(x_i)$: x_i におけるモデル関数の値
δy_i : y_i の誤差
N - 変数の数 : 自由度 ν

- この値はカイ二乗分布と呼ばれる確率密度関数に従う

χ^2 を最小にする理由

- 最も尤もらしいモデル関数は、測定されたデータ値の分布が最も生じやすい関数のはずである
  - 各データ点の誤差（ばらつき）は正規分布に従うとする
  - 各データ点の値が出る確率の積が、手元の標本になる確率になると見なす

$$
\begin{aligned}
\text{Prob.} &\propto \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\delta y_i^2}} \exp\left[-\frac{(y_i - f(x_i))^2}{2\delta y_i^2}\right] \\
&\propto \exp\left[-\sum_{i=1}^{N} \frac{(y_i - f(x_i))^2}{2\delta y_i^2}\right] \\
&= \exp(-\chi^2)
\end{aligned}
$$

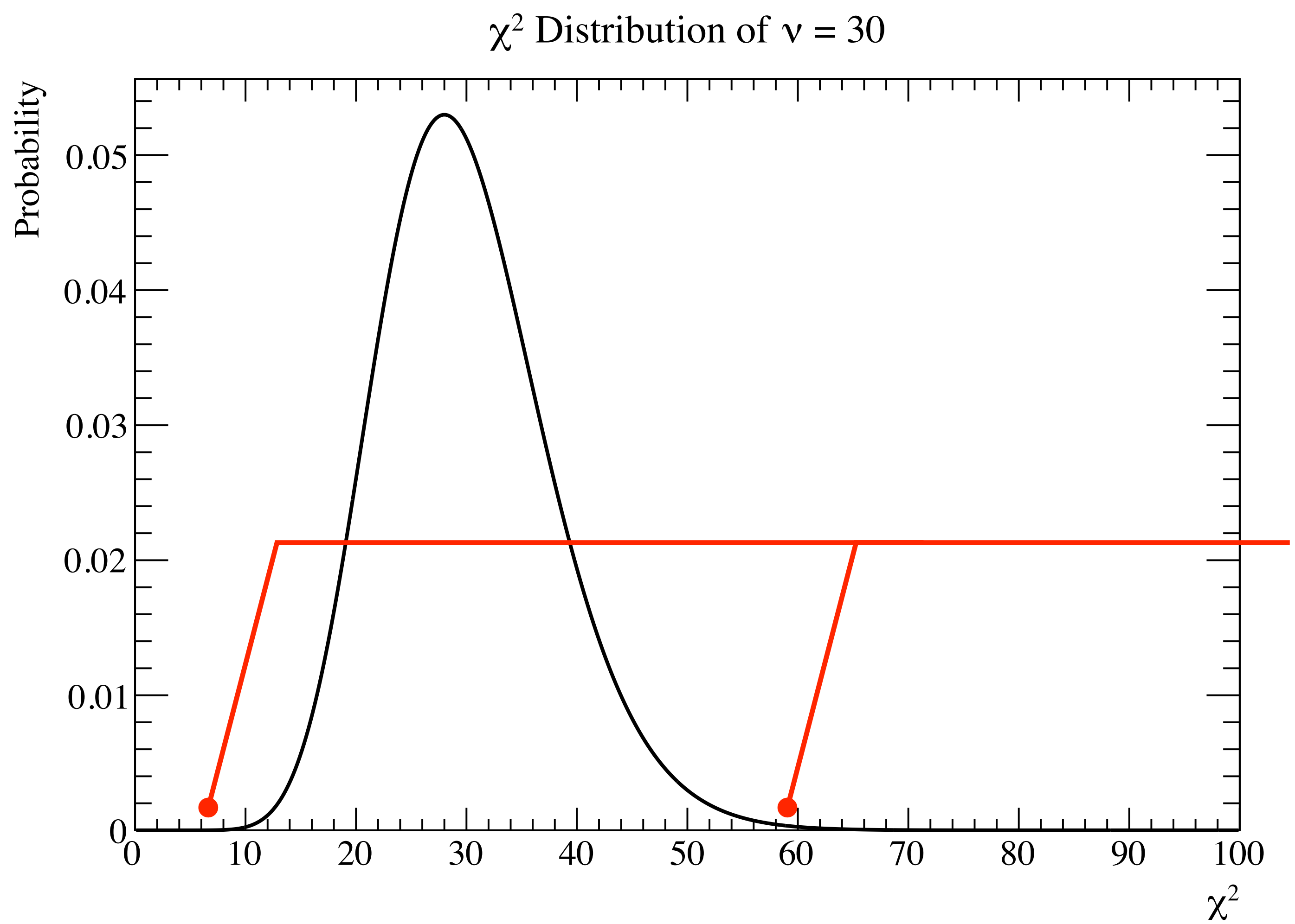
- 結局、 χ^2 を最小にするのが、確率最大になる

カイ二乗分布

- 自由度 ν のカイ二乗の値は、カイ二乗分布に従う

$$P_\nu(\chi^2) = \frac{(\chi^2)^{\nu/2-1} e^{-\chi^2/2}}{\Gamma(\nu/2) 2^{\nu/2}}$$

カイ二乗分布と p 値



このあたりに来ると
確率としてありえない
$p < 0.01$ や $p > 0.99$
くらいの場合、誤差の
評価が正しいか要確認

```
$ root
root [0] TF1* pdf = new TF1("pdf", "ROOT::Math::chisquared_pdf(x, [0], 0)", 0,
100)
root [1] pdf->SetTitle("#chi^{2} Distribution of #nu = 30;#chi^{2};Probability")
root [2] pdf->SetParameter(0, 30)
root [3] pdf->SetNpx(500)
root [4] pdf->Draw()
root [5] TMath::Prob(27.25, 30)
(Double_t) 0.610115
```

❶ カイ二乗分布の 1 次元関数 TF1 を作る

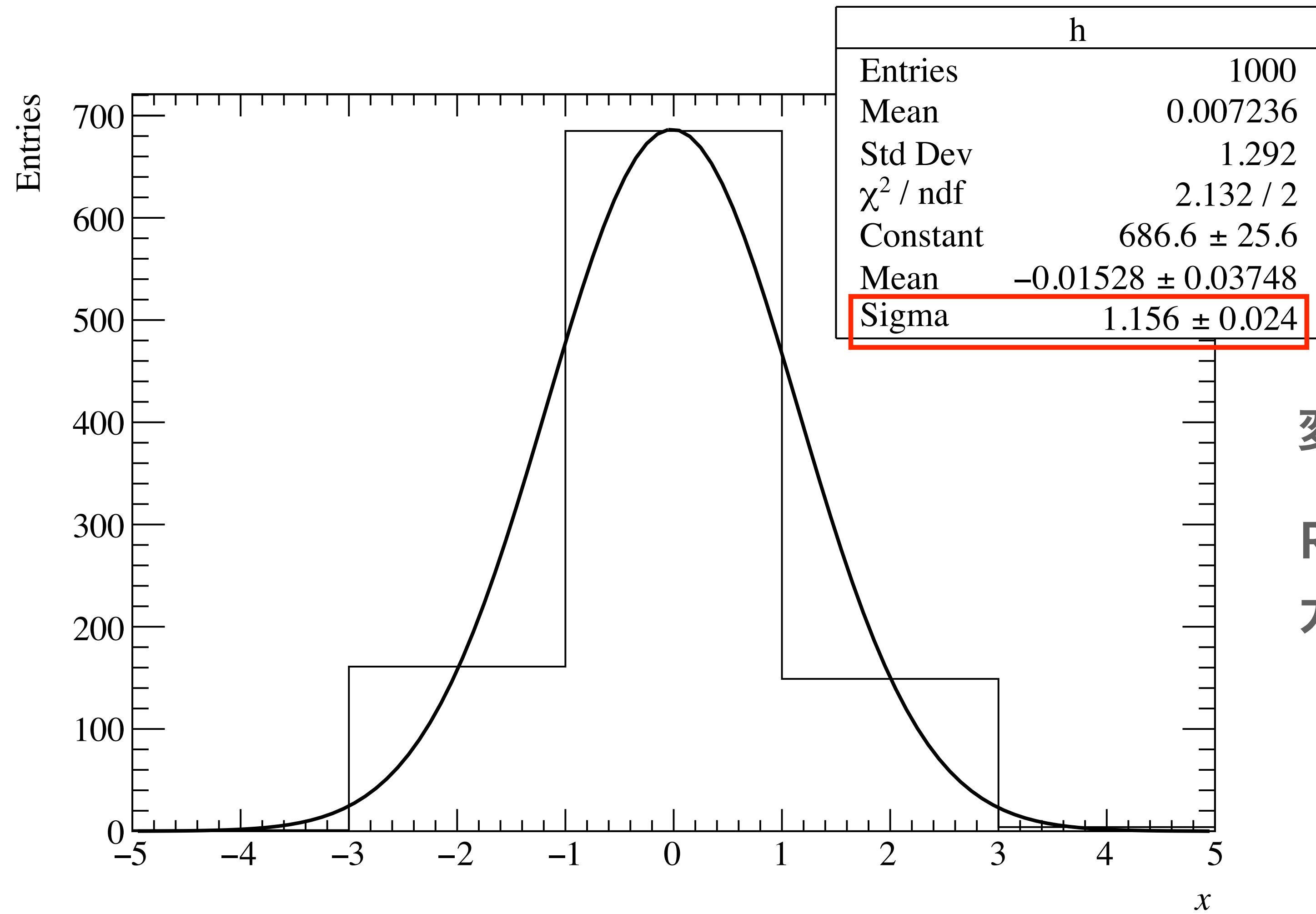
❷ 自由度 ν = 30 に設定

❸ TF1 の点数を増やし表示を滑らかに（本質的でない）

❹ 確率の計算

ν = 30、$\chi 2$ = 27.25 の場合、p = 0.61

モデル関数に比べてビン幅が広過ぎる場合

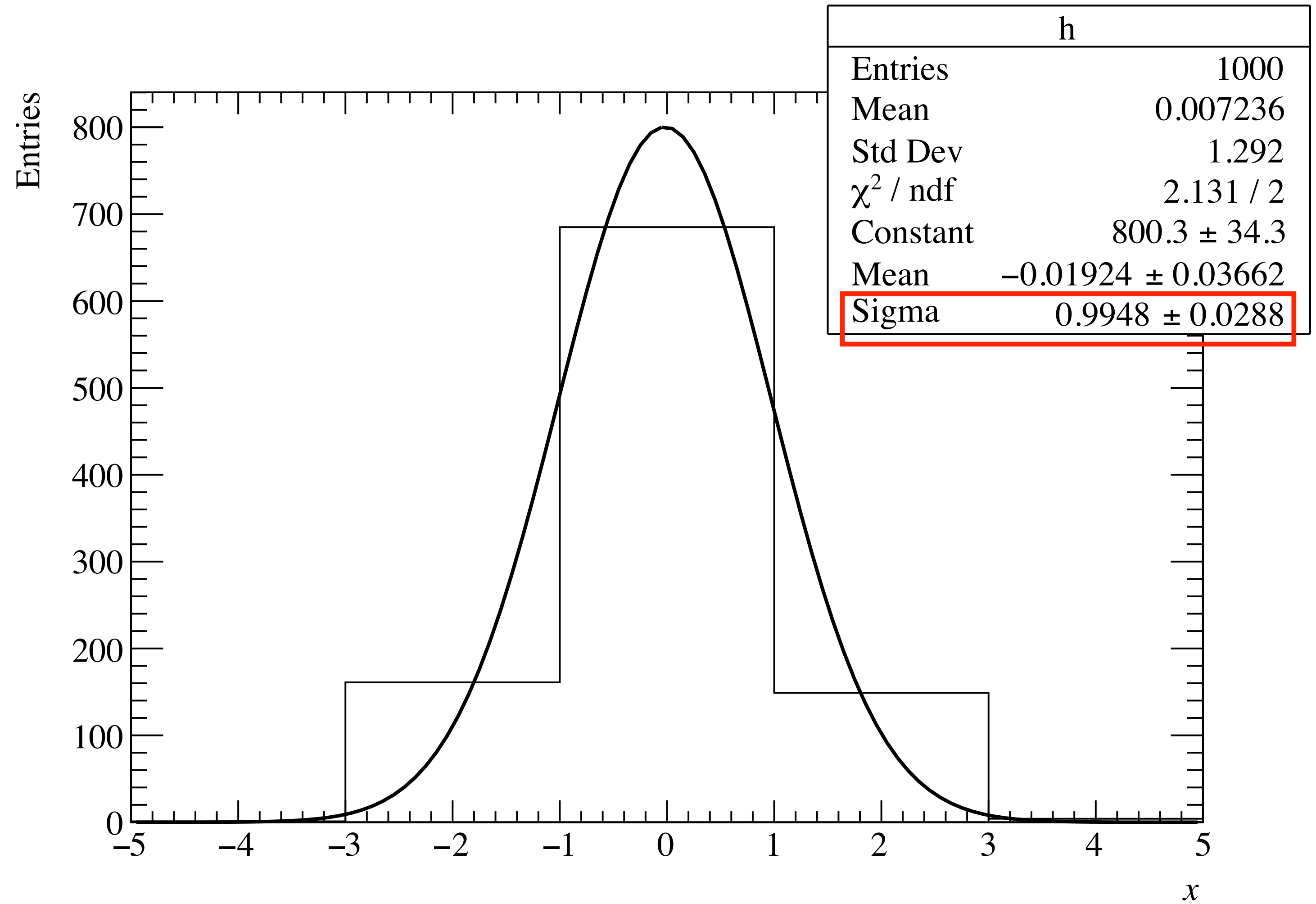


変数の推定を誤る！

ROOT がビンの中心値でカイ二乗を計算するため

```
root [0] TH1D* hist = new TH1D("h", ";#it{x};Entries", 5, -5, 5)
root [1] hist->FillRandom("gaus", 1000)
root [2] hist->Fit("gaus")
 FCN=2.13212 FROM MIGRAD    STATUS=CONVERGED      52 CALLS          53 TOTAL
                     EDM=2.37573e-07    STRATEGY= 1      ERROR MATRIX ACCURATE
  EXT PARAMETER                                  STEP         FIRST
  NO.   NAME      VALUE            ERROR          SIZE      DERIVATIVE
   1  Constant     6.86581e+02   2.55989e+01   1.87885e-02  -1.43368e-05
   2  Mean        -1.52834e-02   3.74843e-02   3.22360e-05   8.88105e-03
   3  Sigma        1.15649e+00   2.36229e-02   4.99586e-06  -1.09181e-01
```

“i” (integral) オプションを使う

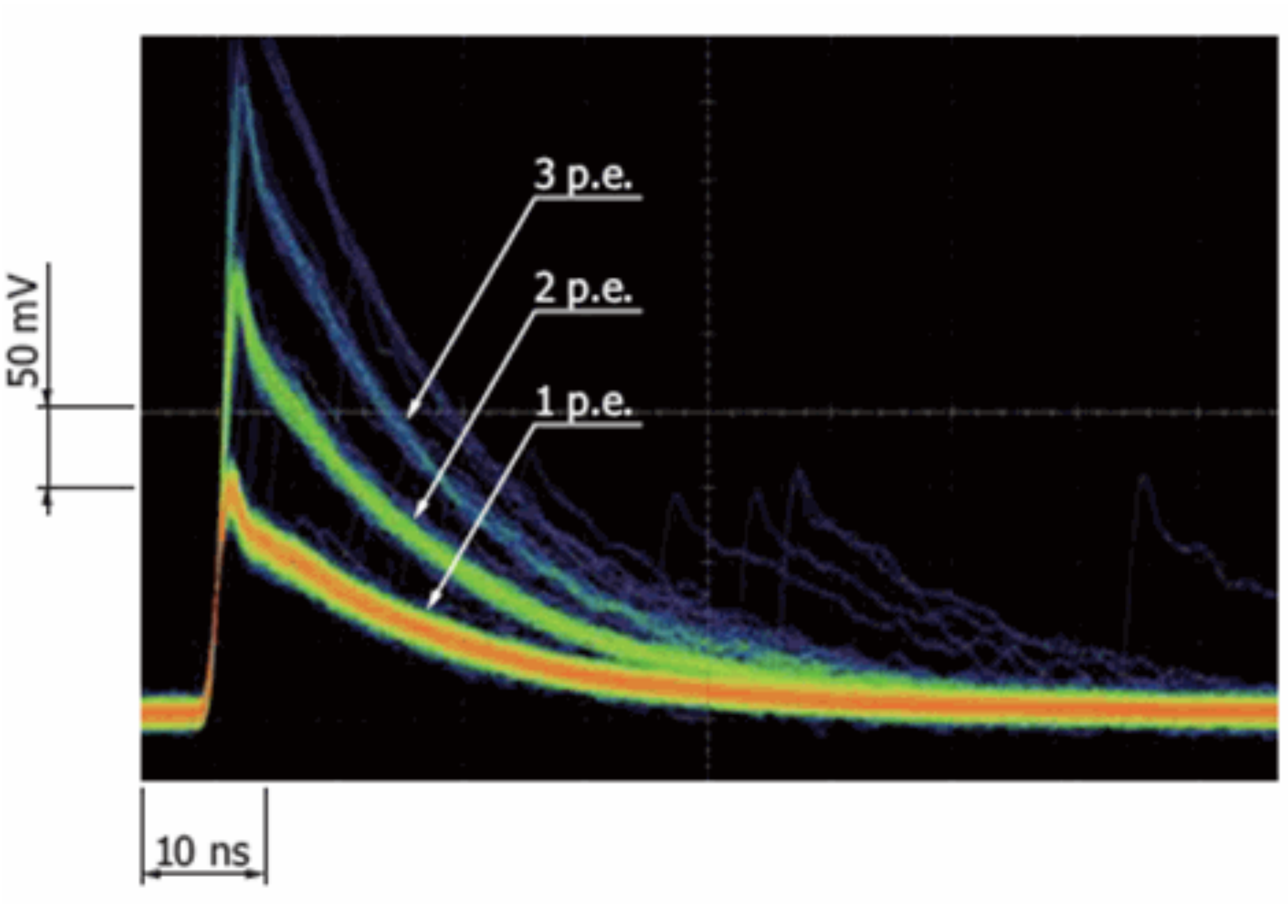



```
root [0] TH1D* hist = new TH1D("h", ";#it{x};Entries", 5, -5, 5)
root [1] hist->FillRandom("gaus", 1000)
root [2] hist->Fit("gaus", "i")
 FCN=2.13123 FROM MIGRAD    STATUS=CONVERGED      104 CALLS         105 TOTAL
                     EDM=2.22157e-07    STRATEGY= 1      ERROR MATRIX ACCURATE
  EXT PARAMETER                                  STEP         FIRST
  NO.   NAME      VALUE            ERROR          SIZE      DERIVATIVE
   1  Constant     8.00322e+02   3.43377e+01   2.18847e-02  -1.06589e-05
   2  Mean        -1.92391e-02   3.66159e-02   3.15299e-05  -1.19143e-03
   3  Sigma        9.94826e-01   2.88088e-02   5.67273e-06  -9.54913e-02
```

“i” を追加

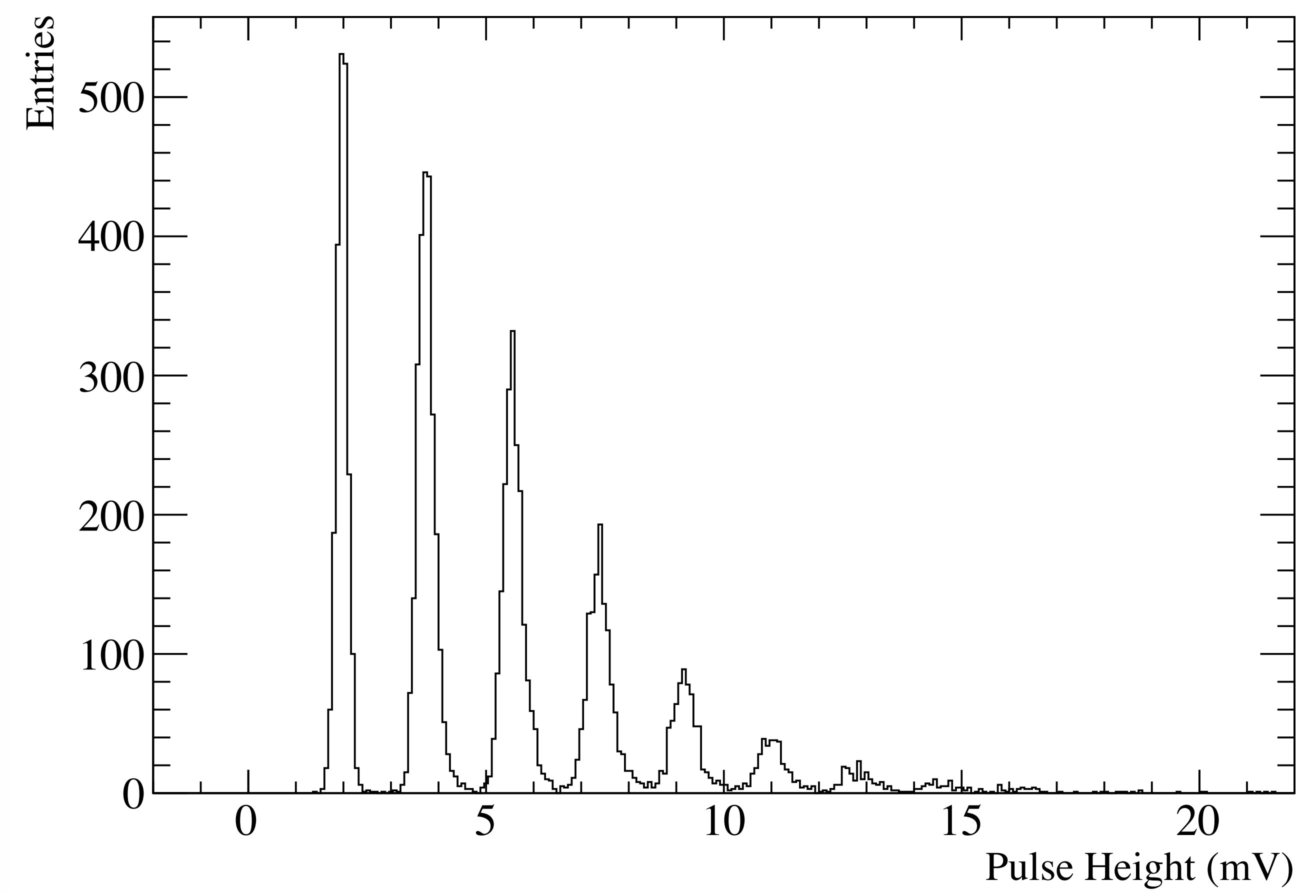
実験室におけるデータ例

正規分布でのフィット例



半導体光検出器の出力波形例
（浜松ホトニクス）

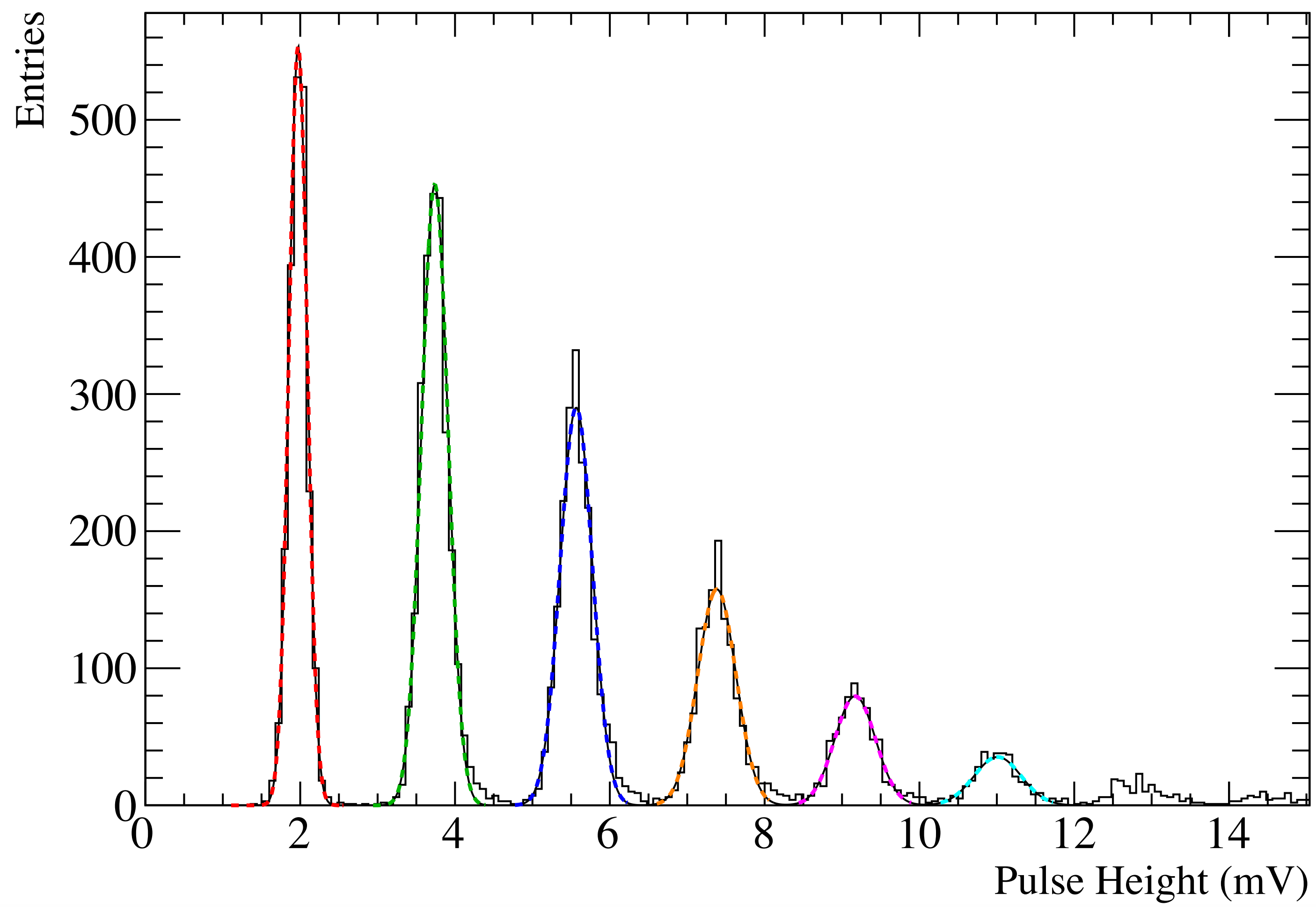
http://www.hamamatsu.com/us/en/community/optical_sensors/sipm/physics_of_mppc/index.html



半導体光検出器の出力波高分布例
（データ提供：日高直哉）

- 光検出器の出力電荷や波高分布は、正規分布でよく近似できる場合が多い
- 半導体光検出器の場合、光電変換された光電子数に比例して波高が綺麗に分かれる
- 光電子数分布や利得（gain）の評価に正規分布でのフィット

複数の正規分布によるフィットの例




```
$ root
root [0] .x MppcFit.C
```

なにをやっているか

```
void MppcFit() {
  TFile file("../misc/MPPC.root");
  gROOT->cd();
  TH1* h = (TH1*)file.Get("pulseheight")->Clone();
  file.Close();

  const Double_t kRoughHeight = 16.5 / 9.; // ~16.5 (mV) at 9 p.e.
  const Int_t kNPeaks = 6;
  TF1* gaus[kNPeaks];

  std::string fit_string = "";

  for (Int_t i = 0; i < kNPeaks; ++i) {
    gaus[i] = new TF1(Form("g%d", i), "gaus", (i + 0.6) * kRoughHeight,
                      (i + 1.4) * kRoughHeight);
    gaus[i]->SetLineColor(i + 2);
    gaus[i]->SetLineStyle(2);

    if (i != 0) {
      fit_string += "+";
    }
    fit_string += Form("gaus(%d)", i * 3);
  }
```

なにをやっているか

```
  TF1* total = new TF1("total", fit_string.c_str(), 1., 17.);
  total->SetLineWidth(1);
  total->SetLineStyle(1);
  total->SetNpx(1000);

  for (Int_t i = 0; i < kNPeaks; ++i) {
    h->Fit(gaus[i], i == 0 ? "R" : "R+");
    for (Int_t j = 0; j < 3; ++j) {
      Double_t p = gaus[i]->GetParameter(j);
      total->SetParameter(i * 3 + j, p);
      if (j == 1) {
        total->SetParLimits(i * 3 + j, p - 0.5, p + 0.5);
      } else if (j == 2) {
        total->SetParLimits(i * 3 + j, p * 0.5, p * 1.5);
      }
      h->Fit(total, "R+");
    }
  }

  h->Fit(total, "i");

  h->Draw();
  h->GetXaxis()->SetRangeUser(0, 15);

  for (Int_t i = 0; i < kNPeaks; ++i) {
    for (Int_t j = 0; j < 3; ++j) {
      Double_t p = total->GetParameter(i * 3 + j);
      gaus[i]->SetParameter(j, p);
    }
    gaus[i]->Draw("l same");
  }
}
```

第 2 回のまとめ

- ヒストグラムとは何か
- TH1 を使った ROOT でのヒストグラムの例
- 正規分布
- カイ二乗分布と確率
- ROOT での 1 次元ヒストグラムのフィッティング

- 分からなかった箇所は、各自おさらいしてください