

- ・本資料の著作権、文責は著者に帰属し、所属機関を代表したり、機関の意見を表明するものではありません。
- ・学校、研究機関の教育、研究目的であれば自由に使用することができます。報告なく改変、再配布可能です。
ただし、明記されている引用先の著作権に配慮してください。
- ・ただし使用者は誤りや誤字を報告する義務があります。

ROOT講習会

第3回資料

ヒストグラム その2



Keita Mizukoshi (Tohoku Univ. RCNS)



今回のテーマ

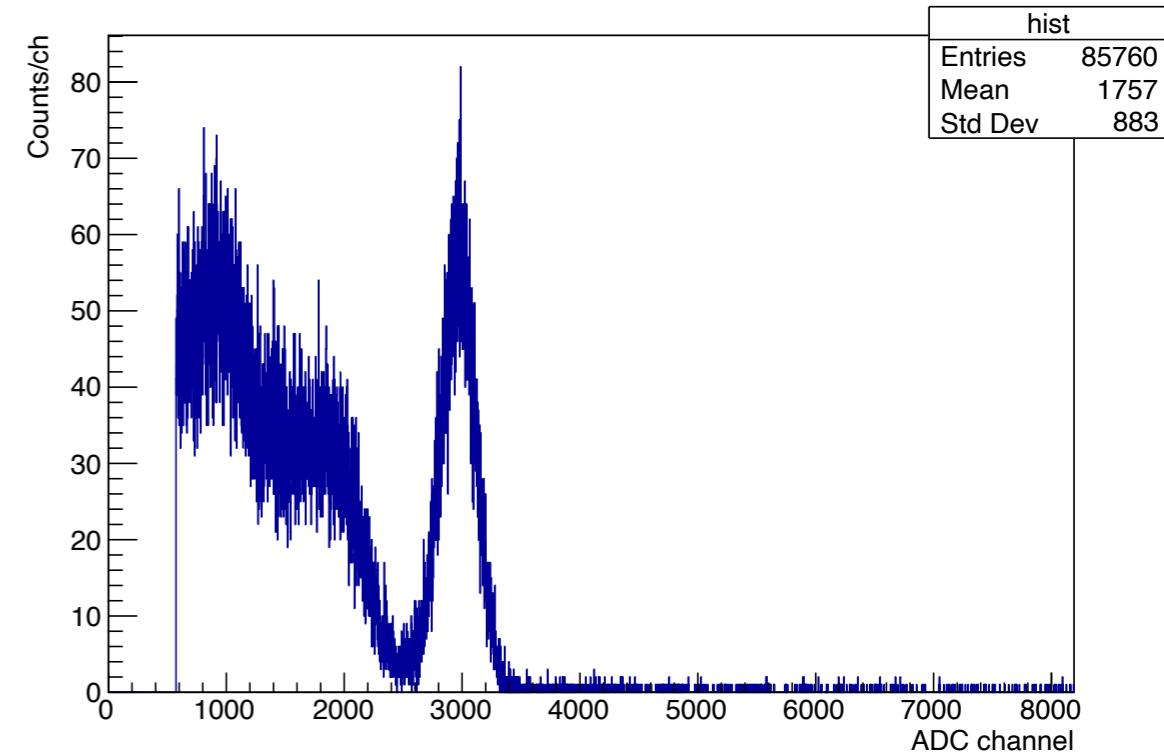
- 第2回でヒストグラムの概念は理解した.
- 今回はやり切れなかったところを頑張る.
- ROOT・解析の小技を覚えていい感じの図を書く.

もう少し物理の実験らしい状況で練習

- CsIシンチレータとPINフォト

ダイオードでCs-137 γ 線源をあてた
データを取得した。

- ADC値の分布が得られた。
- ADC値の範囲 (0–8191)
- データの場所: root_lecture/macros/
mzks/scintillator_exp/data/data1.txt
- さっきの例を応用して自力で
ヒストグラムを表示するマクロをかく
 - 時間とります。



- hagure_expディレクトリから
scintillator_expディレクトリに移動
- macrosディレクトリをつくって
さっきの内容を参考にマクロをかく
- データを念のため目で確認
- Bin数や最大, 最小の値に注意

適切なBin幅

- Bin幅が細かすぎて上下に振れてみづらいので
Bin幅を広げたい
- やり方1: ヒストグラムを作る時にBin数を小さく
設定する.
- やり方2: クラスのメソッド, Rebin()をつかって周
囲のBinをそれぞれまとめる.

```
24
25 hist->Rebin(8); //これで8Binをあたらしい1Binに
26
27 hist->SetTitle(";ADC channel;Counts/ch");
28 hist->Draw();
```

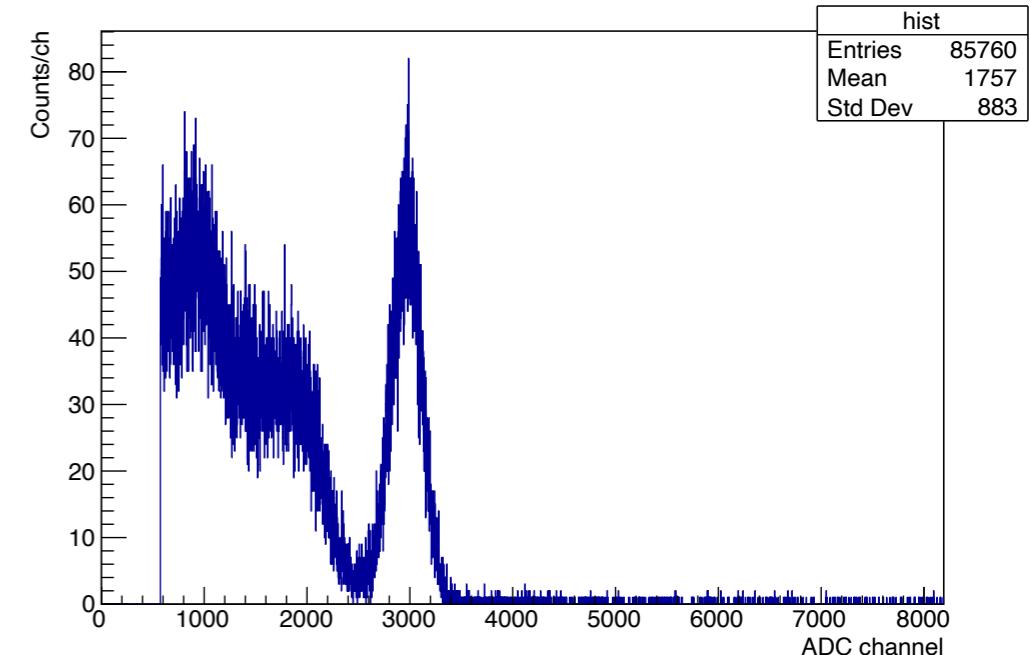
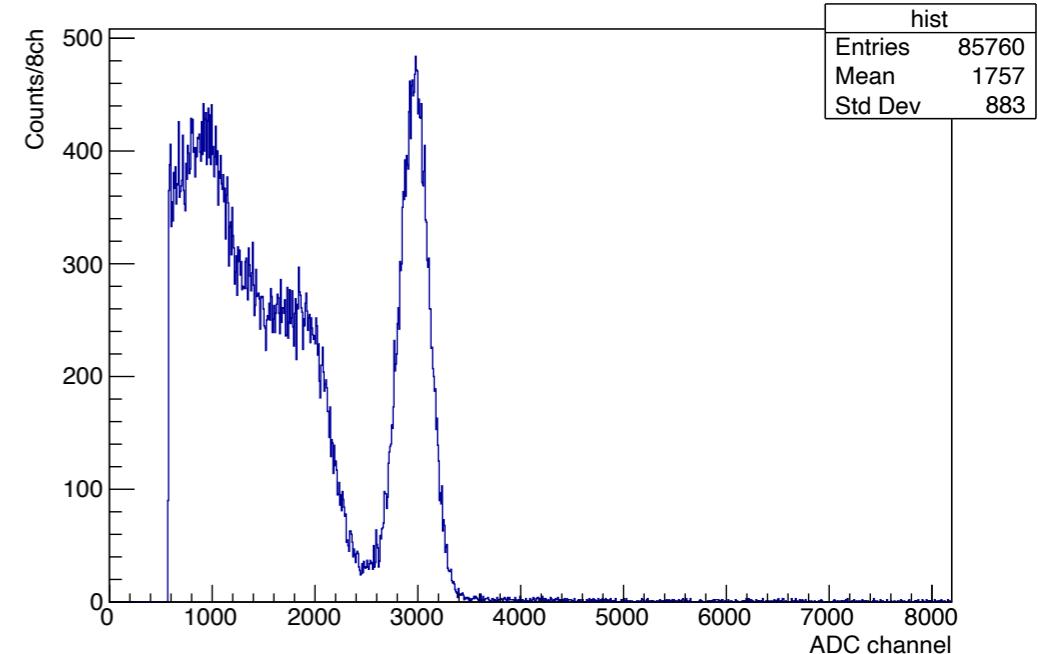


図 再掲



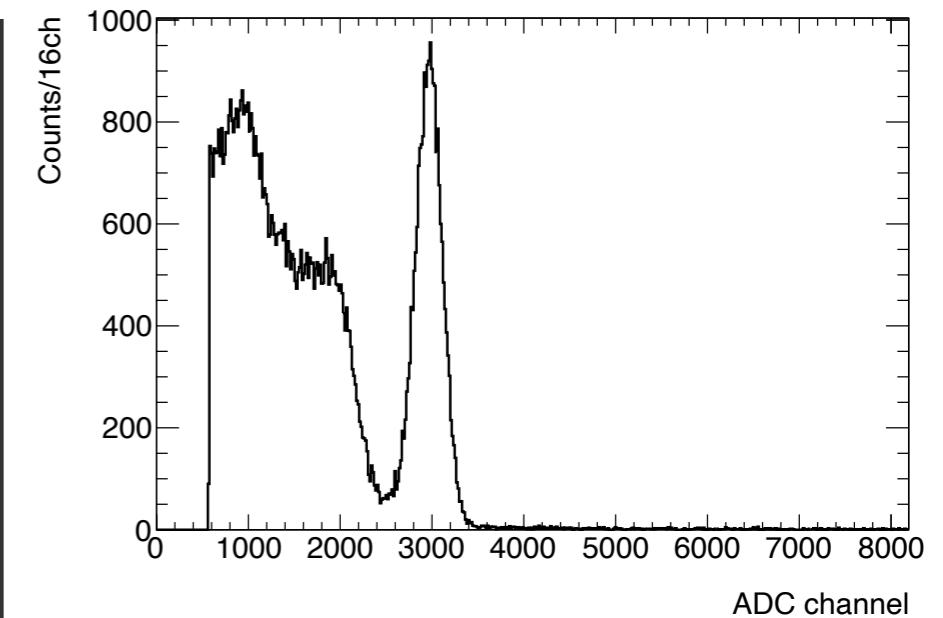
- どんな状況でも使えるBin幅決定アルゴリズムはない.
- 解析する人のセンスが要求される.

Rebin(8)したヒストグラム

マクロ例

```

1 // -----
2 // draw_hist2.C -- macro
3 // Author: K. Mizukoshi
4 // Date : May 12 2020
5 // -----
6
7 int draw_hist2(){
8
9   gROOT->SetStyle("ATLAS");
10
11 // Define Range and Num of bin of histogram
12 const int MCACh = 8192;
13 const double HistMin = 0.;
14 const double HistMax = 8192.;
15
16 TH1D* hist = new TH1D("hist", "hist", MCACh, HistMin-0.5, HistMax-0.5);
17
18 double BufferValue;
19 ifstream ifs("../data/data1.txt");
20 while(ifs >> BufferValue){
21   hist->Fill(BufferValue);
22 }
23
24 hist->Rebin(16);
25
26 hist->SetTitle(";ADC channel;Counts/16ch");
27 hist->Draw();
28
29 return 0;
30 }
```



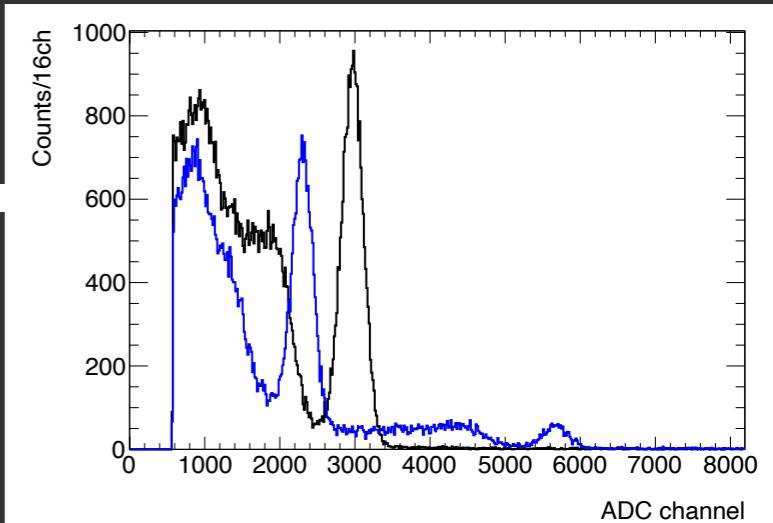
ヒストグラム重ね書き

- data1.txtとdata2.txtのヒストグラムを一つの図に表示する.
- 2つヒストグラムをつくり,1つ目は普通にDraw(), 2つ目はDraw("same");とする.
- 線が同じ色で表示されてしまうので, SetLineColor(kBlue);などをつかってどちらかの線の色を変える.
 - kWhite, kBlack, kGray, kRed, kGreen, kBlue, kYellow, kMagenta, kCyan, kOrange, kSpring, kTeal, kAzure, kViolet, kPinkなどがつかえる
 - ""でくくる必要がないことに注意

```
30     hist1->SetTitle("ADC channel;Counts/16ch");
31     hist1->Draw();
32     hist2->SetLineColor(kBlue);
33     hist2->Draw("same");
```

マクロ例(あまりよくない方法)

```
// draw_hists_not_good.C
7 int draw_hists_not_good(){
8
9 gROOT->SetStyle("ATLAS");
10 const int MCACH = 8192;
11 const double HistMin = 0.;
12 const double HistMax = 8192.;
13
14 TH1D* hist1 = new TH1D("hist1", "hist1" , MCACH, HistMin-0.5, HistMax-0.5);
15 TH1D* hist2 = new TH1D("hist2", "hist2" , MCACH, HistMin-0.5, HistMax-0.5);
16
17 double BufferValue1;
18 ifstream ifs1("../data/data1.txt");
19 while(ifs1 >> BufferValue1){
20     hist1->Fill(BufferValue1);
21 }
22 double BufferValue2;
23 ifstream ifs2("../data/data2.txt");
24 while(ifs2 >> BufferValue2){
25     hist2->Fill(BufferValue2);
26 }
27
28 hist1->Rebin(16);
29 hist2->Rebin(16);
30 hist1->SetTitle(";ADC channel;Counts/16ch");
31 hist1->Draw();
32 hist2->SetLineColor(kBlue);
33 hist2->Draw("same");
34 return 0;
35 }
```

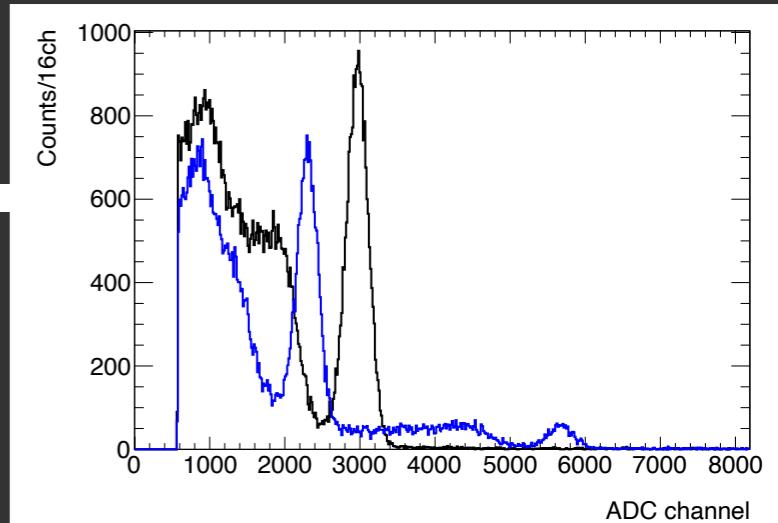


普通に2個ヒストグラムを作り,それぞれ描画する方法
定義, 値詰め, Rebin, Drawをそれぞれ2回ずつ行い,
最後にDraw(), Draw("same")する.

これでも十分動くし,図のクオリティも変わらないが,
データがたくさんになってくることを見据えると
次のページの方法を推奨する.

よいマクロ例

```
// draw_hists.C
7 TH1D* make_hist(TString filename = "data1"){
8
9     gROOT->SetStyle("ATLAS");
10    const int MCACh = 8192;
11    const double HistMin = 0.;
12    const double HistMax = 8192.;
13
14    TH1D* hist = new TH1D(filename, filename, MCACh, HistMin-0.5, HistMax-0.5);
15
16    const TString data_dir = "../data/";
17    double BufferValue;
18    ifstream ifs(data_dir + filename + ".txt");
19    while(ifs >> BufferValue){
20        hist->Fill(BufferValue);
21    }
22
23    hist->Rebin(16);
24    hist->SetTitle(";ADC channel;Counts/16ch");
25    //hist->Draw();
26    return hist;
27 }
28
29 int draw_hists(){
30
31    TH1D* hist1 = make_hist("data1");
32    TH1D* hist2 = make_hist("data2");
33
34    hist1->Draw();
35    hist2->SetLineColor(kBlue);
36    hist2->Draw("same");
37
38    return 0;
39 }
```



さっき作ったdraw_hist2.Cを関数にして再利用
引数でファイルの名前を与える様にする.
最後に,できたヒストグラムをreturnして,
外からヒストグラムが利用できる様にする

渡した先にDrawしてもらうのでコメントアウト

ここでmake_hist()関数から返されたヒストグラムを受け取る.
今は手間が増えたと思うかもしれないが,こうしておくとどれだけ
データファイルが増えても数行たすだけで実現できる.
また,ヒストグラムを作るところと,絵を描くところを構造的に
ロジックに従って分離させることができた.

SetLineColor()で線の色を変えて表示する.
Draw("same")で前のヒストグラムを消さずに
重ねがきできる

メソッドでヒストグラムのパラメータを取得する

```
1   cout << "Sample Size      : " << hist->GetEntries() << endl;
2   cout << "Maximum Value   : " << hist->GetMaximum() << endl;
3   cout << "Maximum Bin     : " << hist->GetMaximumBin() << endl;
4   cout << "Mean             : " << hist->GetMean() << endl;
5   cout << "Standard Dev    : " << hist->GetStdDev() << endl;
6   cout << "Mean Error       : " << hist->GetMeanError() << endl;
7
8   cout << "7th Bin content: " << hist->GetBinContent(7) << endl;
```

- お絵かきするだけではない便利なメソッドがある。
- Sample SizeやMeanをメソッド一発で取得できる。
- GetBinContent()で
Binの値そのものを得る
- SetBinContent()もできる

```
> root -l use_methods.C
Tue May 12 09:19:03 2020
root [0]
Processing use_methods.C...
Sample Size      : 39728
Maximum Value   : 224
Maximum Bin     : 731
Mean             : 3127.93
Standard Dev    : 1791.8
Mean Error       : 8.98962
7th Bin content: 0
Info in <TCanvas::MakeDefCanvas>: created default TCanvas
with name c1
(int) 0
root [1]
```

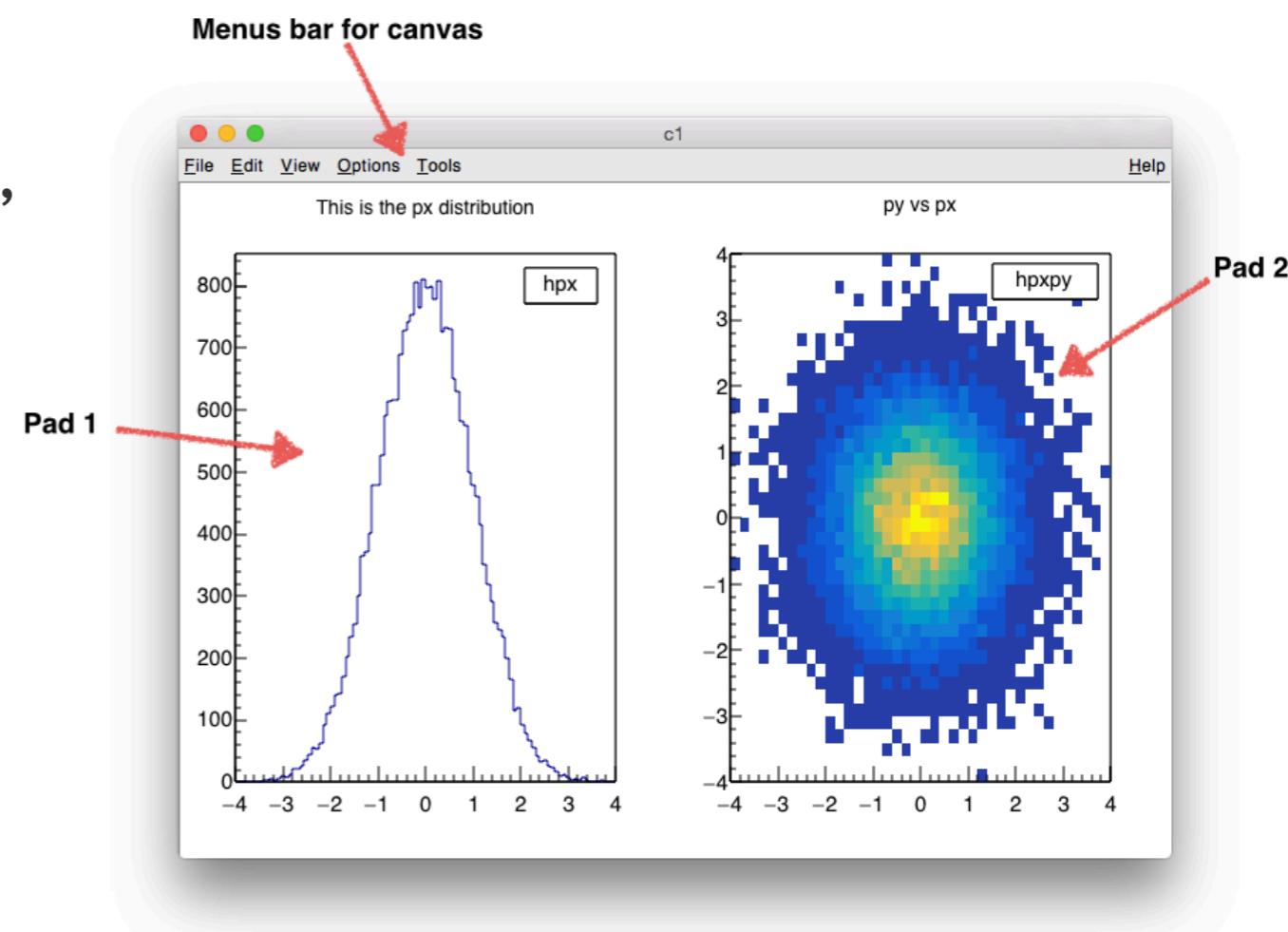
マクロ内で図として保存

- TCanvas* C = new TCanvas();
- <--- ここで色々やる --->
- C->SaveAs("../figures/my_great_figure.pdf");
- これでpdfとして図を保存できる。
- ROOTのメニューからSave...としてもできるが,
面倒なのでマクロを実行した際に自動で保存される
ようにする
- マクロと同じディレクトリに保存するとごちゃごちゃ
してくるので別のディレクトリを作ってそちらに
保存することを推奨。

※前回資料が誤っていました。すみません。

TCanvas

- TCanvas* C = new TCanvas();
- 突然出てきたこれはなんですか?
→ ウィンドウで見えている描画領域全体を表すクラス
- "ヒストグラム"は概念なので画像として保存できない.
- ウィンドウなら保存できる.
- ユーザーが作成していないときは、
ROOTが裏で作成している.
- 明示的に呼びたい場合は自分で
作成しておけばOK
- TPadも勝手に作られる.



プロットを並べる

- プロットを縦や横に並べる.
- TCanvas を作って,
`C->Divide(2,1)` などとすれば
 よい. (横に2分割, 縦に1分割)
- 例: 上にFit, 下に残差を示す
- 今, 使いたいCanvasは,
`C->cd(1)` などと指定.
- C++11以降を使っていれば, auto
 で型推論させられる.
- `C->cd(1)`でどうせTCanvas* の
 型が帰ってくることは自明なの
 で, 計算機に考えさせる機能

```

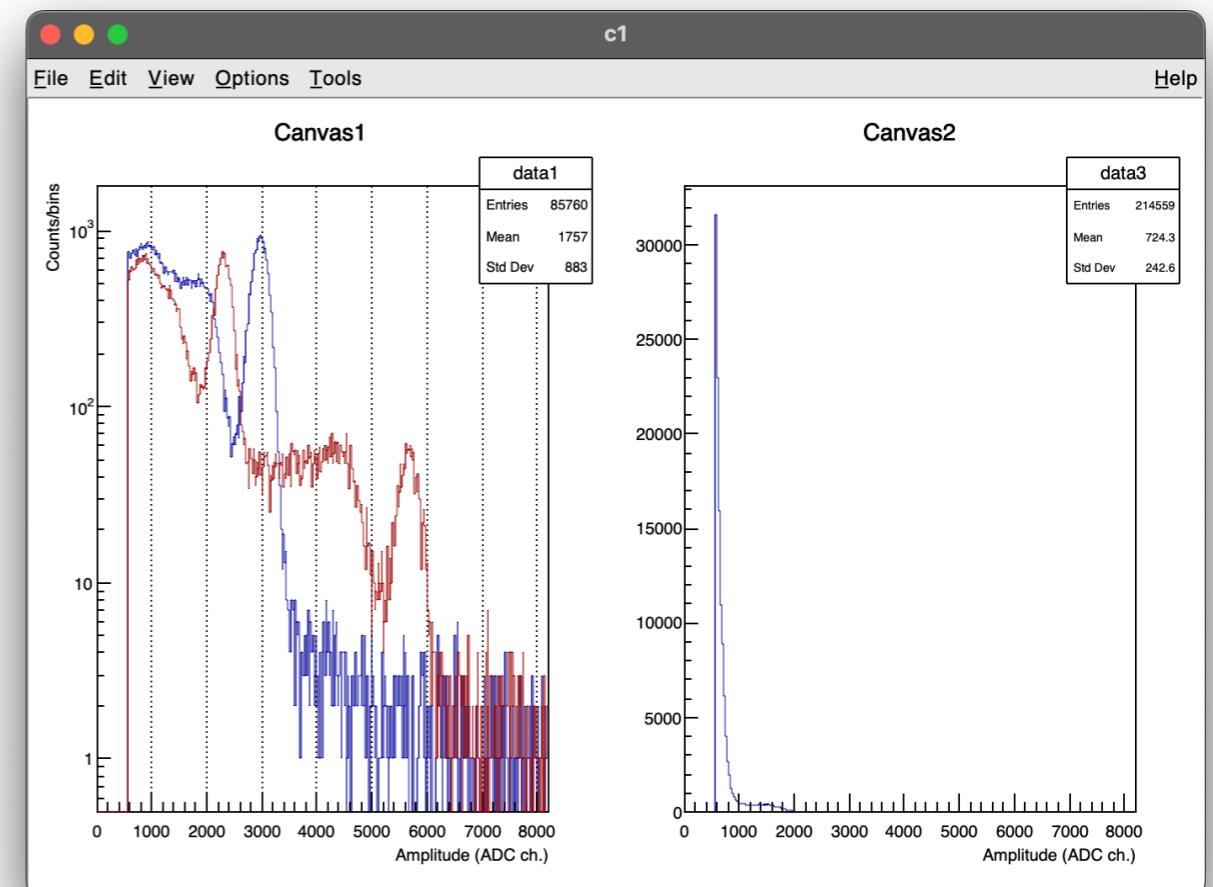
auto C = new TCanvas();

C->Divide(2,1);

auto c1 = C->cd(1);
// ititial canvas id is 1
hist1->Draw();

hist2->Draw("same");

auto c2 = C->cd(2);
hist3->Draw();
  
```



その他の小技

- C->SetLogx() C->SetLogy()
 - 軸をLog scaleに変更する. 特にSetLogy()はよく使う.
- C->SetGridx(), C->SetGridy()
 - グリッド線を引く
- C->BuildLegend()
 - 描かれているものの全てのLegend (凡例) を生成する.
 - 絶妙に使いづらいので, パワポで自分で書いた方がいいです.

すでにBinningされたデータを扱う

- Multi-channel analyzer (MCA) などでスペクトルを取得したデータは、既にBinningされていることがある
 - これまで示してきたのはされていないデータ
 - Binningすれば、データの容量は小さくなるので、出力ファイルとしてはありがち。
- ROOTで読み込みたければ、Binningされた回数の分だけFill()を呼ぶ方法か、Weight付のFill()をする方法か、SetBinContent()で直接値を入れてしまう方法がある。
 - 一長一短。好きな方法でどうぞ。

4	0	17
0	1	12
1	2	9
0	3	8
...	4	...
	...	

Binningされていないデータ
とされているデータ

```
while (ifs >> x >> count){
    for(int i=0;i<count;++i){
        hist->Fill(x);
    }
}
```

Binningされたデータの回数分、
For文で Fill(x)を呼ぶ例

```
while (ifs >> x >> count){
    hist->Fill(x, count);
}
```

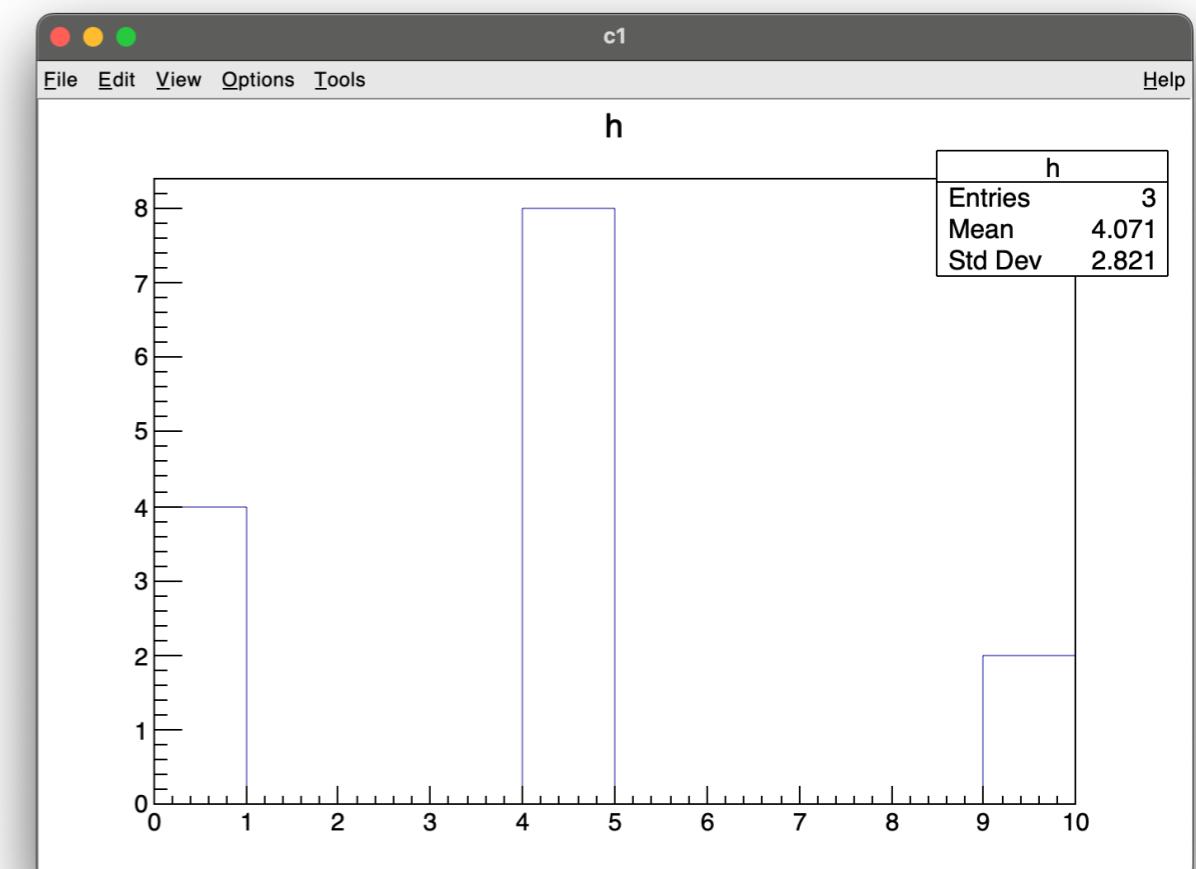
Weight付で Fill(x, w)を呼ぶ例

Binの値を直接取得・編集

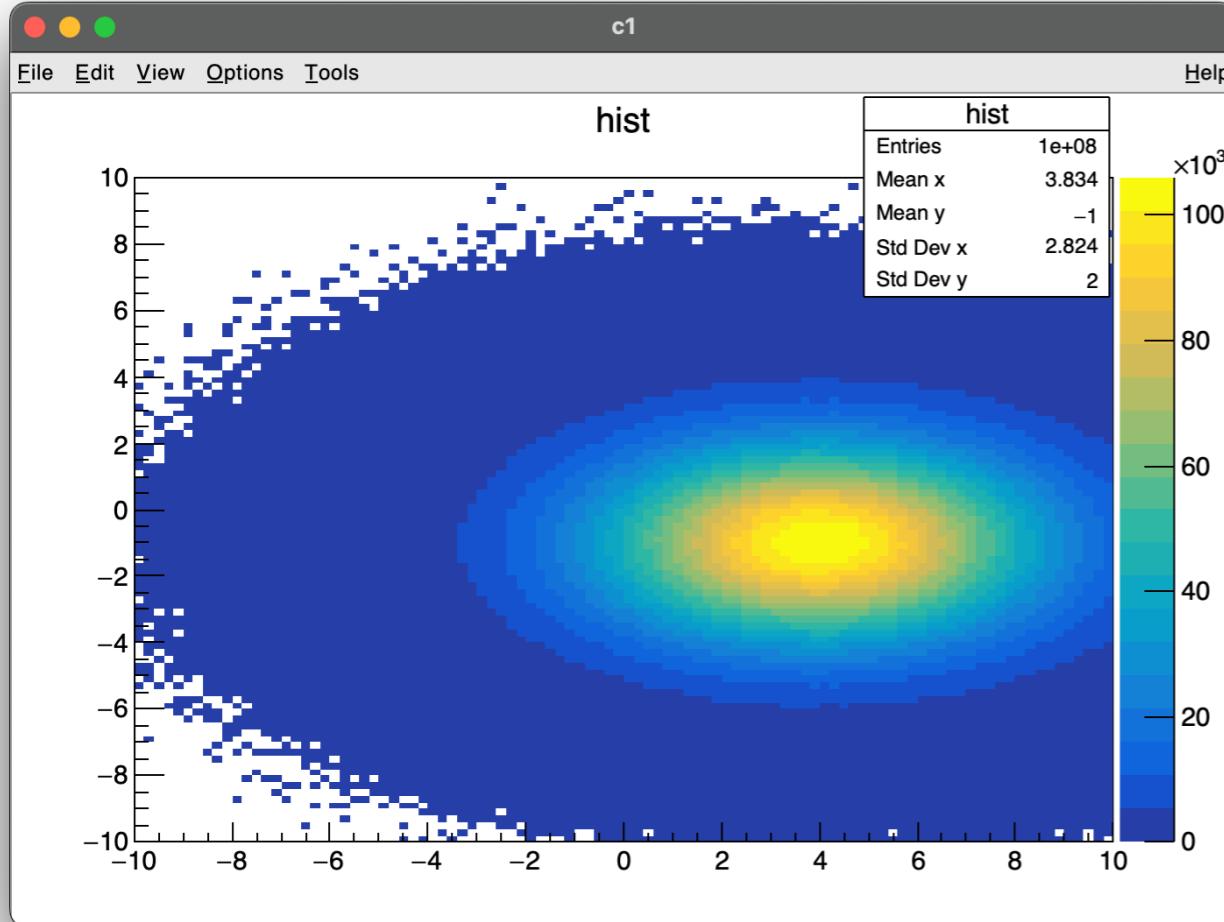
- `hist->SetBinContent(<bin番号>, <値>)` と直接入れることができる。
 - ただし、最初のbin番号は1。
- 1回`SetBinContent`する度に、`Entries`に1足されるので、普通にやったら`Entries != イベント数`になる。嫌なら`Fill()`を使う
- `Error`は変えてくれないので、`SetBinError()`関数を使う
- 特定のBinの値が知りたければ`GetBinContent(<bin番号>)`
- 特定のx軸の値が含まれるBin番号を調べたいときは`FindBin(値)`

```
void mca(){
    auto hist =
        new TH1D("h", "h", 10, 0, 10);

    hist->SetBinContent(1, 4);
    hist->SetBinContent(5, 8);
    hist->SetBinContent(10, 2);
    hist->Draw();
}
```



2次元ヒストグラム



```

void hist_2d(){

const Int_t nbins_x = 100;
const Double_t min_bin_x = -10.;
const Double_t max_bin_x = 10.;

const Int_t nbins_y = 100;
const Double_t min_bin_y = -10.;
const Double_t max_bin_y = 10.;

auto hist = new TH2D("hist", "hist",
nbins_x, min_bin_x, max_bin_x,
nbins_y, min_bin_y, max_bin_y);

TRandom r;
for(int i=0;i<100000000;++i){
    double x = r.Gaus(4, 3);
    double y = r.Gaus(-1, 2);
    hist->Fill(x, y);
}

hist->Draw("colz");
}

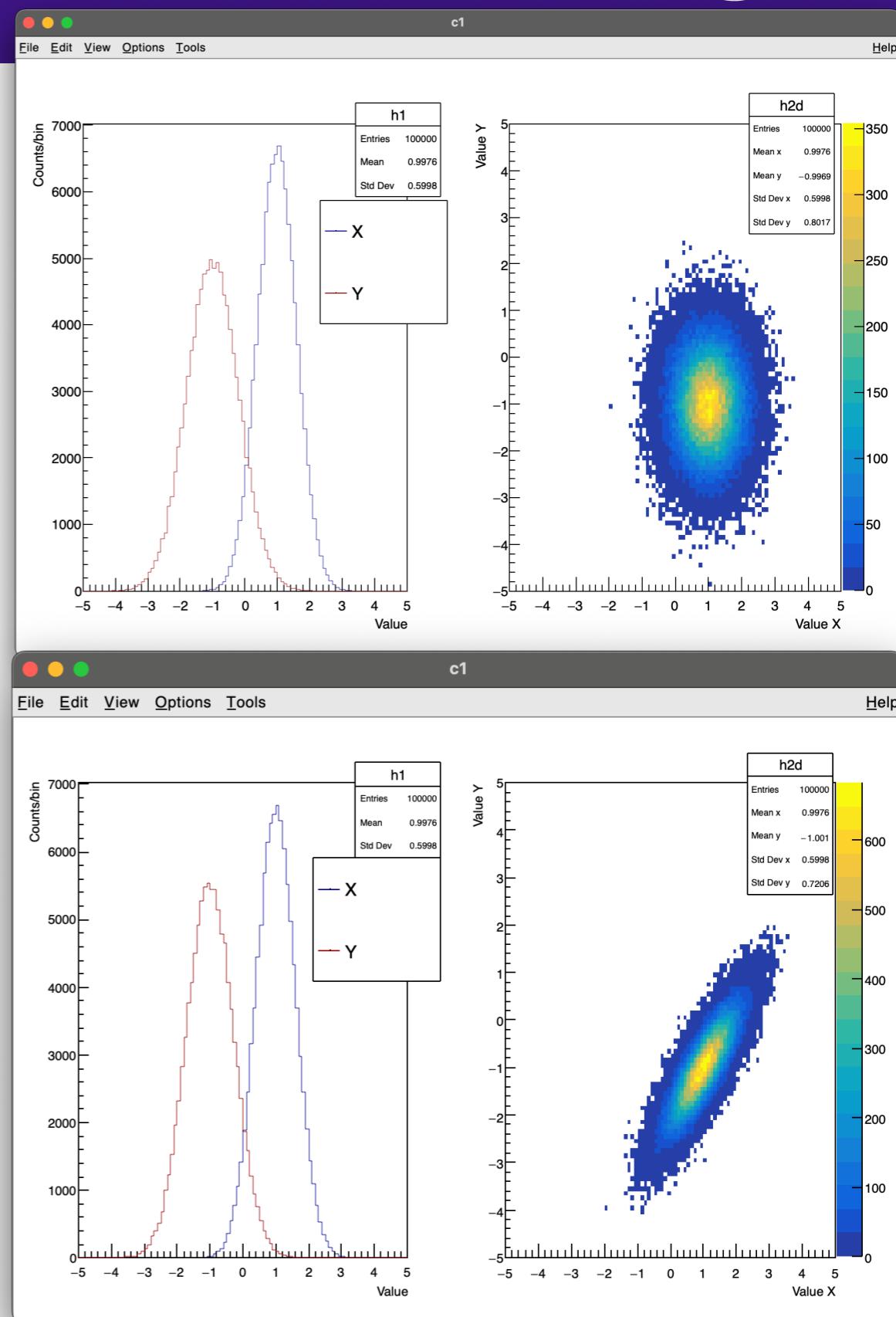
```

- TH1DをTH2Dにして, x, yが必要なところ (コンストラクタ, Draw)で両方書けば良い。



2変数の相関

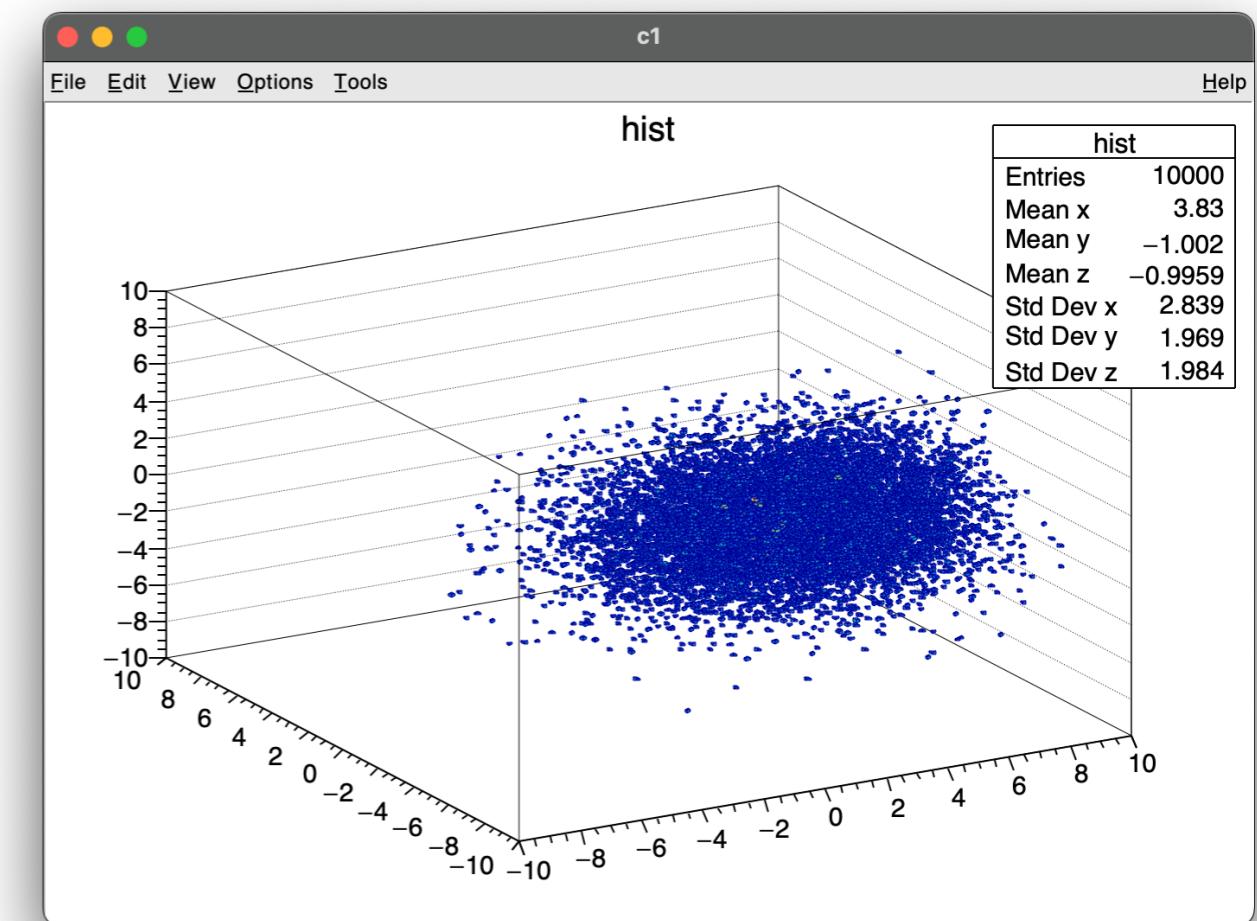
- 1次元ヒストグラムの重ね書きだけでは、変数に相関があるかはわからない。
 - 上下の図で、1次元の分布はほぼ同じ
- 2次元ヒストグラムにして関係を確認する。
 - 2DヒストグラムからProfile, Projectionを作成する
- 実験では、変数が2よりも多くなることが普通。いちいち書くのも面倒なので、Treeに便利な機能が用意されている（後の回で紹介）



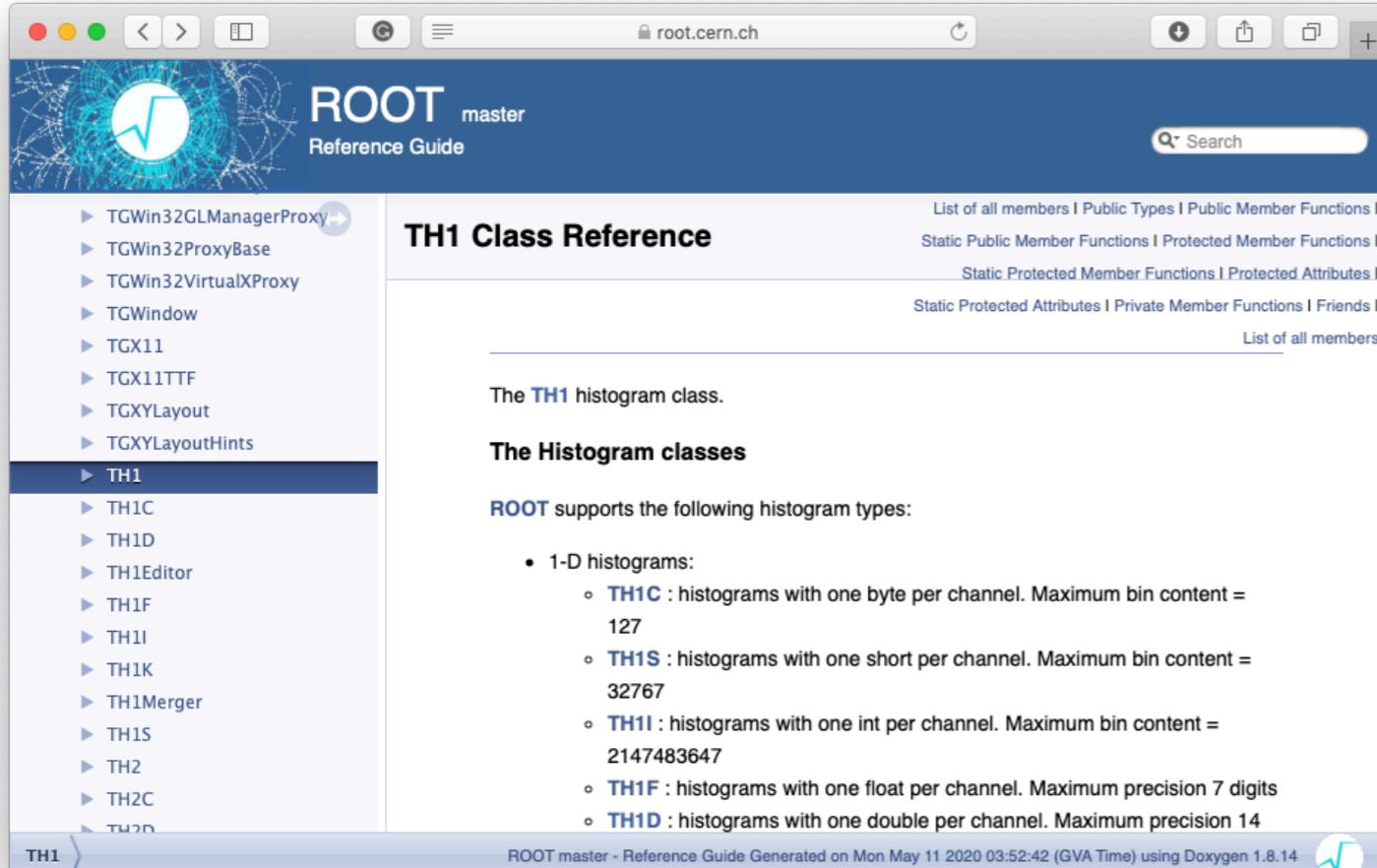
相関なしの例（上）と相関がある例（下）

3次元ヒストグラム

- ・ 同じノリで、3次元ヒストグラムもある（が、シンプルにみづらいのでかなり使いづらい）。
- ・ どちらかといえば、ヒストグラムを作った上で、適当にProjectionして使うためのものだと思う。
- ・ でも、その用途ならTTree（後述）が使える場合が多いので、存在を知っているだけで良いと思う。
- ・ マウスでグルグル回して楽しむ用のものとしては優秀。



クラスリファレンスを見る



ROOT公式リファレンス
<https://root.cern/doc/master/index.html>

最初にここを見る!

- クラスの使い方を知りたい時は公式クラスリファレンスを参照する。
- 適切なクラスを探したり、クラスのメソッドの使い方を探したりできる。
- 微妙に痒いところに手が届かないで、慣れてきたらソースコードを見るようになるかも

次回と今後

- ・ 次回はなんでも質問してくださいの回にしようと思います。
 - ・ 質問を持って集合してください。
- ・ 今後, Fit, Graph, Treeをやって, いったんSeries終了の予定です.
- ・ よろしくお願ひいたします.