ヒストグラム

ROOT講習会2023 第3回

髙橋光成 (名古屋大学ISEE)

自己紹介

- 名前：髙橋光成
- 所属：名古屋大学宇宙地球環境研究所宇宙線研究部
- プロジェクト：CTA, *Fermi*-LAT, MAGIC
- 研究：光検出器（PMT・SiPM）、ガンマ線バースト、活動銀河核

📊データの図示とは

データを図にする理由

数字の羅列ではなぜ分かりにくいか？

1=१=

2=२=

3=३=

- 1,2,3,…,9,0という記号表現とそれが表す記号内容の間には恣意的な関係しかない
  - 互いの関係性が見えづらい
- 問題になっている値の間の関係を目に見える形に落とし込む必要がある
  - 大小関係や差分を紙上の幾何的な配置、距離に変換する
  - 近い値は近くに描き、遠い値は遠くに描く

図への変換 (mapping)
近い値、遠い値とは

- 何を以て値が近い/遠いとするかは描く人が判断する必要がある
  - 例1) 1と100, 1000と2000ではどちらが近い値か？
  - 例2) 0°, 180°, 300°の中で近い値の組み合わせは？

そしてヒストグラムへ
情報を整理する

- 例) 測定を何回も繰り返し、その結果を図示する場合
  - 軸上にプロットするだけではまだ分かりにくい
- 人間は物を数えるのは苦手
- 「数えた結果」も図上に落とし込む
  - ある範囲の値は同等であるとみなす

ROOTのヒストグラム

宇宙線分野の観測データの例

- ROOTでのヒストグラムの作り方は前回学習した
- 例としてガンマ線望遠鏡*Fermi*-LATのデータを用いる
- 2008年に打ち上げられた高エネルギー宇宙ガンマ線観測衛星*Fermi*に搭載
  - 全天の30 MeV–300 GeVガンマ線を観測
  - このエネルギー帯のガンマ線は主に衝撃波などで加速された粒子から放射される
    - 粒子のエネルギーの増加・減少がエネルギー自体に比例し、倍々で起こる
      - エネルギーや時間、個数の比が重要→対数表示が活躍

観測データをヒストグラムにしよう

- root_lecture/macros/tkhs/lat-data/lat_photons_300MeV.dat
- 中身を見てみると、ガンマ線1個ずつの情報が書かれている
- 飛来方向は銀河系の中心・円盤面を基準にした座標と衛星からみて地球の中心（の反対）・自転軸を基準にした座標がある

1 行が1個の光子に対応 →

```
(base) mitsunari@White-CTA-iMac macros % head ../lat-data/allsky_photons.dat
  4931.1074   186.40201    66.69105     110.99175      30.568527             4.5610560374749446E8
  2988.297    57.856243    52.6086      111.2355       331.76807             4.5610560458237594E8
  516.18414   82.95196     67.50891     111.29916      352.37335             4.5610560459165955E8
  442.3429    205.57336    63.7599      115.185455     38.6572               4.561056051064911E8
  1036.0813   42.96566     39.94485     113.50167      314.379               4.561056051262641E8
  778.4724    53.01713     48.3351      111.54488      326.1625              4.5610560537444526E8
  367.00674   184.2203     69.56075     112.504745     27.951687             4.561056057890055E8
  404.6052    45.00639     39.51818     111.96931      314.87454             4.561056061210675E8
  77387.74    168.77509    2.4389622    50.717117      58.118896             4.56105606131219E8
  594.9343    126.68344    74.78837     111.78197      9.015549              4.5610560617372376E8
```

エネルギー [MeV]

飛来方向の銀河座標(L, B)[°]

飛来方向の対地座標(Z, AZ)[°]

到来時刻 [s]

目標

右のようなガンマ線飛来
方向のマップを描いてみる

注意事項

- 今回配布したマクロは虫食いになっています
- /*ここを変えて*/　というコメントになっている部分を正しいコードに置き換えてください
- スライド中では置き換えたものを見せていきます

```
// histogram_1D.C

void histogram_1D(){
    // Construct a 1D histogram
    const Int_t NBIN_ENERGY = 100;
    const Int_t ENERGY_MIN = 0;
    const Int_t ENERGY_MAX = 100000; //100 GeV
    TH1D* hist = /*ここを変えて*/ /*ここを変えて*/(“hist_1D", "Energy;[MeV];[events]", /*ここを変えて*/, /*ここを変えて*/, /*ここを変えて*/);
}
```

単純な可視化

- まずガンマ線光子のエネルギーを1次元ヒストグラムTH1Dに詰めてみる
  - root [0] .x ./histogram_1D.C()

histogram_1D.C

処理系非依存の型Int_t, Double_t等を用いるとエラーを防げる

```
// histogram_1D.C

void histogram_1D(){
    // Construct a 1D histogram
    const Int_t NBIN_ENERGY = 100;
    const Int_t ENERGY_MIN = 0;
    const Int_t ENERGY_MAX = 100000; //100 GeV
    TH1D* hist = new TH1D("hist_1D", "Energy;[MeV];[events]",
NBIN_ENERGY, ENERGY_MIN, ENERGY_MAX);
    // Make the histogram fancy
    hist->SetFillStyle(3001);
    hist->SetFillColor(kBlue);
    hist->SetLineColor(kBlue);
    hist->SetLineWidth(2);

    // Open input file
    ifstream ifs("../lat-data/allsky_photons.dat");
    // Check if input file is open
    if(!ifs.is_open()){
        cout << "Input file is not opened!" << endl;
        return 0;
    }
    // Fill the values to the histogram
    Double_t energy, l, b, zenith, azimuth, time;

    while(ifs >> energy >> l >> b >> zenith >> azimuth >> time){
        hist->Fill(energy);
    }

    // Construct a canvas
    TCanvas* can = new TCanvas("can", "All-sky photons");
    // Draw the histogram
    can->cd();
    hist->Draw();
}
```

物理的意味が分かるように単位などをコメントすると良い

単純な可視化

- まずガンマ線光子のエネルギーを1次元ヒストグラムTH1Dに詰めてみる
  - root [0] .x ./histogram_1D.C()

histogram_1D.C

処理系非依存の型Int_t, Double_t等を用いるとエラーを防げる

```
// histogram_1D.C

void histogram_1D(){
    // Construct a 1D histogram
    const Int_t NBIN_ENERGY = 100;
    const Int_t ENERGY_MIN = 0;
    const Int_t ENERGY_MAX = 100000; //100 GeV
    TH1D* hist = new TH1D("hist_1D", "Energy;[MeV];[events]",
NBIN_ENERGY, ENERGY_MIN, ENERGY_MAX);
    // Make the histogram fancy
    hist->SetFillStyle(3001);
    hist->SetFillColor(kBlue);
    hist->SetLineColor(kBlue);
    hist->SetLineWidth(2);

    // Open input file
    ifstream ifs("../lat-data/allsky_photons.dat");
    // Check if input file is open
    if(!ifs.is_open()){
        cout << "Input file is not opened!" << endl;
        return 0;
    }
    // Fill the values to the histogram
    Double_t energy, l, b, zenith, azimuth, time;

    while(ifs >> energy >> l >> b >> zenith >> azimuth >> time){
        hist->Fill(energy);
    }

    // Construct a canvas
    TCanvas* can = new TCanvas("can", "All-sky photons");
    // Draw the histogram
    can->cd();
    hist->Draw();
}
```

物理的意味が分かるように単位などをコメントすると良い

希少事象への着目

- 0–10^5 MeVの範囲でプロットしているが、1番目のビンのカウントが多すぎて分かりにくい
- 縦軸を対数にする
  - `root [1] can->SetLogy()`
- 高いエネルギーにも少数ながらガンマ線が分布していることが分かる
  - 0と1の差が10000と100000の差に負けず劣らず重要な場合がある

区間スケールの再定義

- 宇宙素粒子分野ではしばしばエネルギーを対数で取り扱う
- 横軸（区間）も対数にしてみる
  - root [2] can->SetLogx()
- 両対数表示にした時に直線をなす=<u>べき分布を持つ</u>ことが分かる
- ビンの幅が著しく異なるせいでまだ見にくい

ビン区切りの再定義

- そもそも軸のスケールを対数にするならビンの区切りも対数を基準にするべきでは？
  - 「同等とみなす値の幅」を差の代わりに比で考える
- ビンを定義する変数Energyを$\log_{10}$Energyに変える

histogram_1D_logE.C

対数をとった値で軸を定義する

```
    // Construct a 1D histogram
    const Int_t NBIN_ENERGY = 100;
    const Int_t ENERGY_MIN = 300; //100 MeV
    const Int_t ENERGY_MAX = 100000; //100 GeV
    TH1D* hist = new TH1D("hist_1D",
"Energy;log_{10}(Energy[MeV]);[events]",
NBIN_ENERGY, log10(ENERGY_MIN), log10(ENERGY_MAX));
```

対数をとった値を詰める

```
    // Fill the values to the histogram
    Double_t energy, l, b, zenith, azimuth, time;
    while(ifs >> energy >> l >> b >> zenith >>
azimuth >> time){
        hist->Fill(log10(energy));
    }
```

ビン区切りの再定義

- そもそも軸のスケールを対数にす

- log10Energyに変える

histogram_1D_logE.C

対数をとった値で軸を定義する

```
    // Construct a 1D histogram
    const Int_t NBIN_ENERGY = 100;
    const Int_t ENERGY_MIN = 300; //100 MeV
    const Int_t ENERGY_MAX = 100000; //100 GeV
    TH1D* hist = new TH1D("hist_1D",
"Energy;log_{10}(Energy[MeV]);[events]",
NBIN_ENERGY, log10(ENERGY_MIN), log10(ENERGY_MAX));
```

対数をとった値を詰める

```
    // Fill the values to the histogram
    Double_t energy, l, b, zenith, azimuth, time;
    while(ifs >> energy >> l >> b >> zenith >>
azimuth >> time){
        hist->Fill(log10(energy));
    }
```

もっと柔軟に

- 横軸の変数はEnergyのままで区切りを柔軟に設定することも可能
  - ビン数+1の長さの配列を用いる

histogram_1D_flexible.C

対数をとった時に等間隔になるように配列に値を入れる

```
    // Define your X-axis flexibly
    const Int_t NBIN_ENERGY = 100;
    const Int_t ENERGY_MIN = 300; //300 MeV
    const Int_t ENERGY_MAX = 100000; //100 GeV
    Double_t xbins[NBIN_ENERGY+1];
    const Double_t LOGE_BINWIDTH = (log10(ENERGY_MAX) -
log10(ENERGY_MIN)) / NBIN_ENERGY; // Divide the X-axis by
an equivalent width in log-space
    for(int ibin=0; ibin<NBIN_ENERGY; ibin++){
        xbins[ibin] = ENERGY_MIN * pow(10, LOGE_BINWIDTH *
ibin);
    }
    xbins[NBIN_ENERGY] = ENERGY_MAX;

    // Construct a 1D histogram
    TH1D* hist = new TH1D("hist_1D", "Energy;Energy[MeV];
[events]", NBIN_ENERGY, xbins);
```

第３引数：ビン数、第４引数：ビン区切りの配列

```
    hist->Draw();
    can->SetLogx();
    can->SetLogy();
```

キャンバスで両軸を対数表示にする

もっと柔軟に

- 横軸の変数はEnergyのままで区

histogram_1D_flexible.C

対数をとった時に等間隔になるように配列に値を入れる

```
    // Define your X-axis flexibly
    const Int_t NBIN_ENERGY = 100;
    const Int_t ENERGY_MIN = 300; //300 MeV
    const Int_t ENERGY_MAX = 100000; //100 GeV
    Double_t xbins[NBIN_ENERGY+1];
    const Double_t LOGE_BINWIDTH = (log10(ENERGY_MAX) -
log10(ENERGY_MIN)) / NBIN_ENERGY; // Divide the X-axis by
an equivalent width in log-space
    for(int ibin=0; ibin<NBIN_ENERGY; ibin++){
        xbins[ibin] = ENERGY_MIN * pow(10, LOGE_BINWIDTH *
ibin);
    }
    xbins[NBIN_ENERGY] = ENERGY_MAX;

    // Construct a 1D histogram
    TH1D* hist = new TH1D("hist_1D", "Energy;Energy[MeV];
[events]", NBIN_ENERGY, xbins);
```

第３引数：ビン数、第４引数：ビン区切りの配列

```
    hist->Draw();
    can->SetLogx();
    can->SetLogy();
```

キャンバスで両軸を対数表示にする

エラーバーを付ける

- 各ビンにおける統計誤差はROOTが自動的に計算してくれる
  - 各ビンでの重みの二乗和の平方根
    - 重みについては次頁
- Drawのオプションで**E**を指定
  - `root [3] hist_1D->Draw("E")`

重みをつける

- カウント数よりもエネルギー流量に着目したい場合がある
- エネルギー流量を図示するには、ヒストグラムの各ビンにエントリを詰める際にエネルギー値で重みをつける
- エラーバーは重みの二乗和の平方根になる
  - 例) 重み1のエントリが10個ならエラーは$\sqrt{10}$、重み2のエントリが5個ならエラーは$\sqrt{20} = 2\sqrt{5}$

histogram_1D_weight.C

```
    // Fill the values to the histogram
    Double_t energy, l, b, zenith, azimuth, time;
    while(ifs >> energy >> l >> b >> zenith >>
azimuth >> time){
        hist->Fill(energy, energy);
    }
```

第2引数が重みになる (省略すると1)

分布と座標系

- 対地座標の天頂角の分布を見てみる
  - root [0] .x ./histogram_1D_zenith.C()
- 天頂角~113°にガンマ線イベントが集中している
  - 対地座標で特定の領域に集中していることは天体由来ではないことを示唆する
  - 正体は“Earthlimb”と呼ばれる地球大気由来のガンマ線

場合分け
詰めるイベントの選別

- Earthlimbガンマ線を除いてエネルギー分布を描き直す
  - if文を使えば良い
  - 条件を満たすかによって重みに1または0をかける手もある

histogram_1D_weight_zenithcut.C

ヒストグラムを2個用意

```
    // Construct a 1D histogram
    TH1D* hist_astro = new TH1D("hist_astro",
"Astronomical;Energy [MeV];EdN [MeV]", NBIN_ENERGY, xbins);
    TH1D* hist_earth = new TH1D("hist_earth", "Earthlimb;Energy
[MeV];EdN [MeV]", NBIN_ENERGY, xbins);
    hist_astro->SetLineColor(kBlue);
    hist_astro->SetLineWidth(2);
    hist_earth->SetLineColor(kRed);
    hist_earth->SetLineWidth(2);
```

if文で場合分けしてFill

```
    // Fill the values to the histogram
    Double_t energy, l, b, zenith, azimuth, time;
    while(ifs >> energy >> l >> b >> zenith >> azimuth >> time){
        if(zenith<100)
            hist_astro->Fill(energy, energy);
        else
            hist_earth->Fill(energy, energy);
    }
```

または、重みに判定式を掛け算する

```
    // Fill the values to the histogram
    Double_t energy, l, b, zenith, azimuth, time;
    while(ifs >> energy >> l >> b >> zenith >> azimuth >> time){
        hist_astro->Fill(energy, energy * (zenith<100));
        hist_earth->Fill(energy, energy * (zenith>=100));
    }
```

複数のヒストグラムを重ねて描く

- 複数のヒストグラムを１枚の図に描くにはTHStackを用いる

histogram_1D_weight_zenithcut.C

THStackを作り、そこにTH1DをAddで追加

```
  THStack *hstack = new THStack("hstack", "Energy
spectra;Energy [MeV];EdN [MeV]");
  hstack->Add(hist_astro);
  hstack->Add(hist_earth);

    // Construct a canvas
    TCanvas* can = new TCanvas("can", "All-sky
photons");
    // Draw the histogram
    can->cd();
    hstack->Draw("NOSTACK"); // Draw without
stacking histograms
```

個々のTH1Dの代わりにTHStackをDrawする

積み重ねない場合はオプションで”NOSTACK”

複数のヒストグラムを重ねて描く

- 複数のヒストグラムを１枚の図に描くにはTHStackを用いる

histogram_1D_weight_zenithcut.C

THStackを作り、そこにTH1DをAddで追加

```
  THStack *hstack = new THStack("hstack", "Energy
spectra;Energy [MeV];EdN [MeV]");
  hstack->Add(hist_astro);
  hstack->Add(hist_earth);

    // Construct a canvas
    TCanvas* can = new TCanvas("can", "All-sky
photons");
    // Draw the histogram
    can->cd();
    hstack->Draw("NOSTACK"); // Draw without
stacking histograms
```

個々のTH1Dの代わりにTHStackをDrawする
積み重ねない場合はオプションで”NOSTACK”

📊２次元ヒストグラム

TH2D

- ２次元ヒストグラムはTH2D
- オプションにより色々なスタイルで描画できる

histogram_2D.C

```
    // Construct a 2D histogram
    const Int_t NBIN_L = 30;
    const Int_t L_MIN = -180;
    const Int_t L_MAX = 180;
    const Int_t NBIN_B = 15;
    const Int_t B_MIN = -90;
    const Int_t B_MAX = 90;

    TH2D* hist = new TH2D("hist_2D", "Sky map;-L[deg];B[deg]",
NBIN_L, L_MIN, L_MAX, NBIN_B, B_MIN, B_MAX);
```

X軸に加えてY軸を設定

```
    // Construct a canvas
    TCanvas* can = new TCanvas("can", "All-sky photons", 1800,
600);
    // Divide the canvas into 6 pads
    can->Divide(3, 2);
    gStyle->SetOptStat(0);

    // Draw the histogram with some different style options
    can->cd(1);
    hist->Draw("COLZ");
    can->cd(2);
    hist->Draw("TEXT");
    can->cd(3);
    hist->Draw("COLZ TEXT");
    can->cd(4);
    hist->Draw("BOX");
    can->cd(5);
    hist->Draw("CANDLE");
    can->cd(6);
    hist->Draw("LEGO");
```

色々な描画オプションが使える

分かりやすく表示する

Sky map
B[deg]
-L[deg]
Sky map
B[deg]
-L[deg]
Sky map
B[deg]
-L[deg]
Sky map
B[deg]
-L[deg]
Sky map
B[deg]
-L[deg]
Sky map
B[deg]
-L[deg]

1次元ヒストグラムへの投影

- ProjectionY, ProjectionXで投影が出来る

histogram_2D_projection.C

Y軸方向へ投影 (TH2のY軸はTH1のX軸になる)

```
    // Draw the histogram and its projections
    can->cd(1);
    hist->Draw("COLZ");
    can->cd(2);
    TH1D* hist_projY = (TH1D*)hist->ProjectionY("hist_projY");
    hist_projY->Draw("E");
    can->cd(3);
    TH1D* hist_projX = (TH1D*)hist->ProjectionX("hist_projX", 10,
12);
    hist_projX->Draw("E");
```

Y軸ビン10番目から12ビン目までをX軸方向へ投影

プロファイルで見る

histogram_2D_profile.C

- ２次元ヒストグラムは全体的な傾向・推移を掴むには向いていない場合もある
  →TH2からTProfileを作る

- 平均値と誤差（デフォルトでは標準誤差）を表示

オプションsで平均±標準偏差を表示

```
    // Draw the histogram and its profiles
    can->cd(1);
    hist->Draw("COLZ");
    can->cd(3);
    TProfile* profileX_stddev = (TProfile*)hist->ProfileX("profileX_stddev",1,-1,"s");
    profileX_stddev->Draw();
    can->cd(4);
    TProfile* profileX_stderr = (TProfile*)hist->ProfileX("profileX_stderr");
    profileX_stderr->Draw();
```

デフォルトは平均±標準誤差

インタラクティブに見る

- スライスするビンを変えながらインタラクティブに投影させることができる

図法を変える

histogram_2D_AITOFF.C

```
// Draw the histogram
can->cd();
hist->Draw("AITOFF");
hist->SetMaximum(200);
gStyle->SetPalette(62);
```

Aitoff図法を指定

値の最大値を制限

色を宇宙っぽくする

- いい感じになった
- 銀河面のほか、いくつかの活動銀河核、パルサー星雲などが見える

おわりに

- やったこと
  - 図示、ヒストグラムを用いる動機
  - １次元・２次元ヒストグラムを作成してデータを表示する
  - 軸に自分の欲しいスケール・ビン区切りを設定する
  - 必要な情報を見やすく表示させる
- 質問のススメ
  - ROOTのヒストグラム、ツリーの機能は極めて豊富で、使い方の情報が見つけにくい場合もあるのでどんどん質問すると良いです。
- YMAPへの参加、ROOT講習会の運営・講師としての参加も歓迎します!!
  http://www.icrr.u-tokyo.ac.jp/YMAP/join.html