高エネルギー宇宙物理学のための ROOT 入門

– 第 6 回 –

奥村 曉

名古屋大学 宇宙地球環境研究所

2017 年 6 月 8 日

名古屋大学 NAGOYA UNIVERSITY

ISEE Institute for Space-Earth Environmental Research

oxon@mac.com

numpy を入れてください（事前通告なくすみません）

```
$ python
>>> import numpy
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ImportError: No module named numpy

$ sudo easy_install pip
$ sudo pip install numpy

$ python
>>> import numpy
```

❶ もし numpy が入っていなかったら

❷ まず pip を入れる（easy_install は多分入ってるはず）

❸ pip を使って numpy を入れる

❹ numpy を import できるか確認

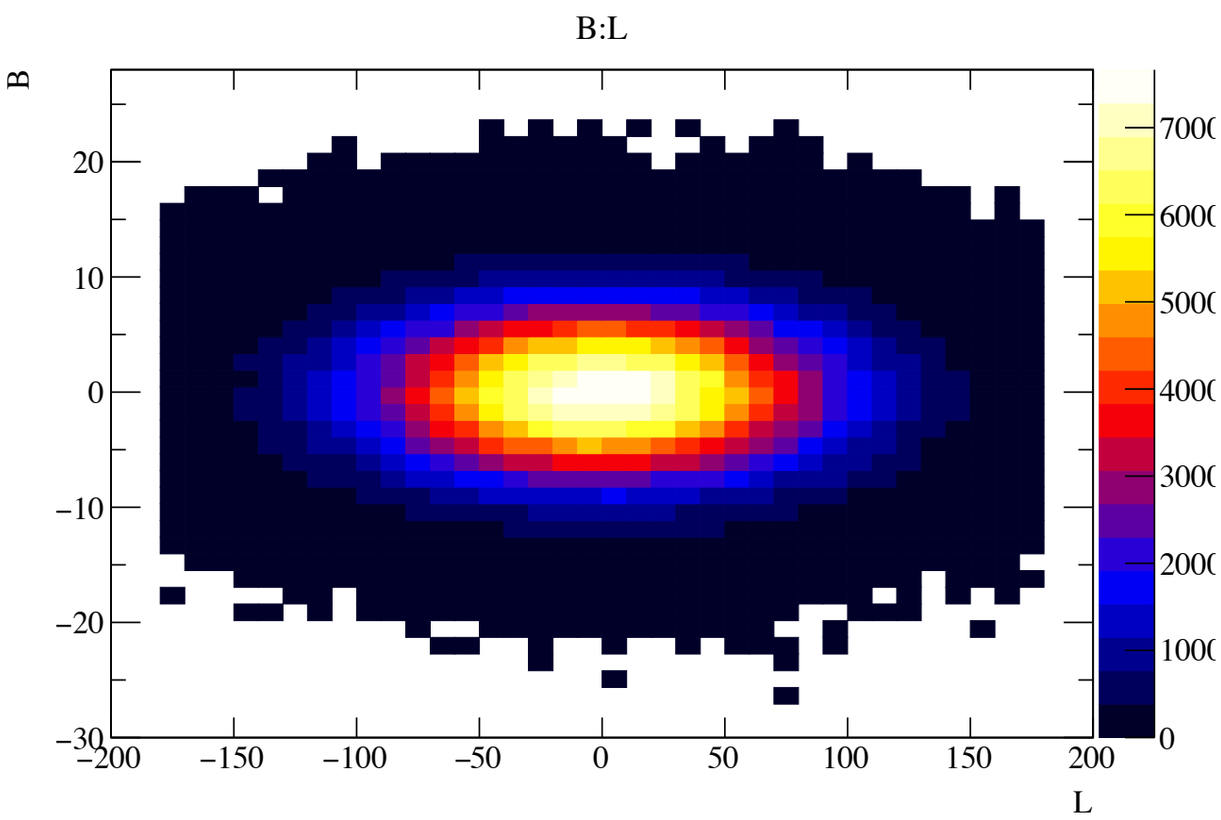
TTree の続き

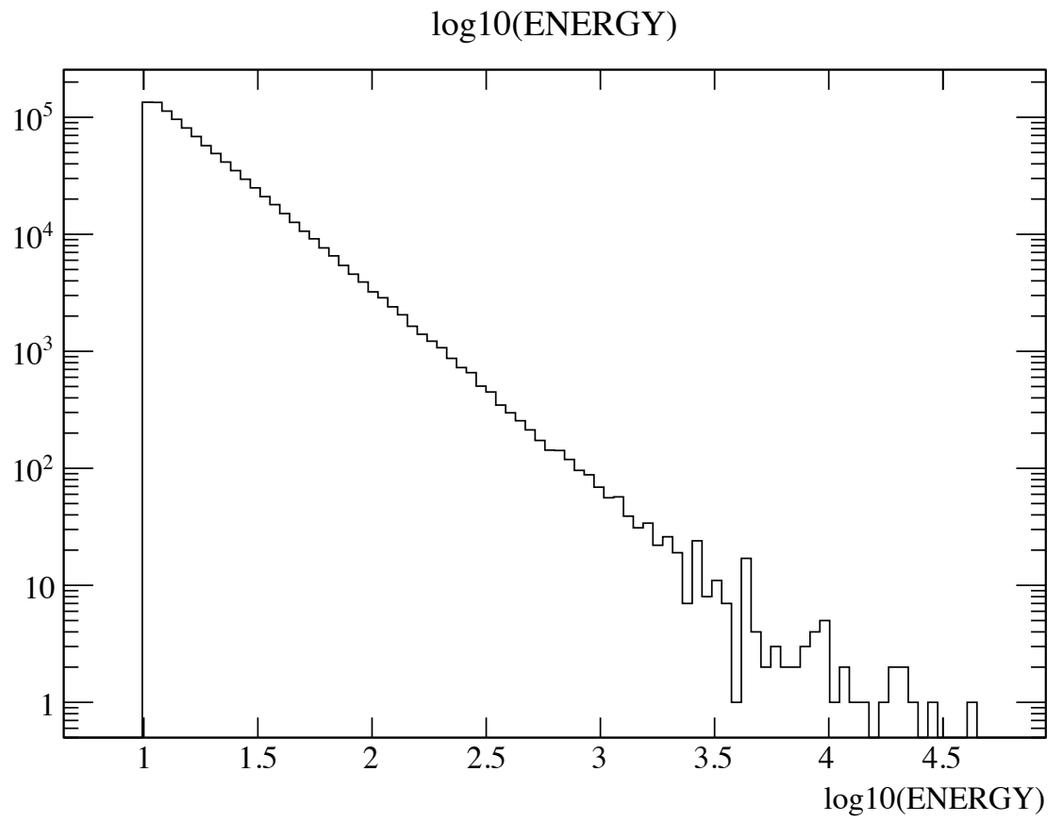
まずは TNtuple から

TNtuple とは

- Q. なんで TTree じゃなくて TNtuple からやるの?
  A. そっちのほうが簡単だから
- TNtuple は TTree の派生クラス
- TTree には int でも double でも ROOT のクラスで
  も詰められるが、TNtuple は float しか詰められない
  - やれることが非常に限られる
  - その分、TTree（に近い概念）を理解するのが楽
- 使いどころ
  - 手早く解析したいとき
  - データの型を気にしなくて良く、データ構造が単純なとき

単純な例（前回の Fermi/LAT データのようなもの）






```
root [0] TNtuple nt("nt", "test", "ENERGY:L:B")
root [1] TF1 f1("f1", "x**(-2.7)", 10, 1000000)
root [2] while(nt.GetEntries() < 1000000){
root (cont'ed, cancel with .@) [3] float e = f1.GetRandom();
root (cont'ed, cancel with .@) [4] float l = gRandom->Gaus(0, 60);
root (cont'ed, cancel with .@) [5] float b = gRandom->Gaus(0, 5);
root (cont'ed, cancel with .@) [6] if(abs(l) <= 180 && abs(b) <= 90) nt.Fill(e, l, b);
root (cont'ed, cancel with .@) [7]}
root [8] nt.Draw("B:L", "", "colz")
root [9] nt.Draw("log10(ENERGY)")
root [10] gPad->SetLogy(1)
```

❶ TNtuple 作成。第三引数は float の変数名一覧。

❷ -2.7 乗の冪に従う乱数発生用の一変数関数

❸ 詰めたい変数値を
イベントごとに代入し

❹ Fill することでイベントを増やす

❺ 後は前回の TTree と同様に遊ぶ

TTree の読み書き（1）

```
$ root misc/lat_photon_weekly_w009_p302_v001.root
root [1] photons->Print()
******************************************************************************
*Tree    :photons   : LAT PASS8 Photons                                       *
*Entries :   177778 : Total =        27471504 bytes  File  Size =   27453414 *
*        :          : Tree compression factor =   1.00                       *
******************************************************************************
*Br    0 :ENERGY    : ENERGY[1]/F                                            *
*Entries :   177778 : Total  Size=     713624 bytes  File Size  =     712860 *
*Baskets :       23 : Basket Size=      32000 bytes  Compression=   1.00     *
*.............................................................................*
*Br    1 :RA        : RA[1]/F                                                *
*Entries :   177778 : Total  Size=     713516 bytes  File Size  =     712768 *
*Baskets :       23 : Basket Size=      32000 bytes  Compression=   1.00     *
*.............................................................................*
*Br    2 :DEC       : DEC[1]/F                                               *
*Entries :   177778 : Total  Size=     713543 bytes  File Size  =     712791 *
*Baskets :       23 : Basket Size=      32000 bytes  Compression=   1.00     *
*.............................................................................*
(省略)
$ curl -O https://raw.githubusercontent.com:443/akira-okumura/RHEA-Slides/master/photons/
lat_photon_weekly_w009_p302_v001_extracted.root
$ root lat_photon_weekly_w009_p302_v001_extracted.root
root [1] photons->Print()
******************************************************************************
*Tree    :photons   :                                                         *
*Entries :   166224 : Total =         2001714 bytes  File  Size =    1830019 *
*        :          : Tree compression factor =   1.09                       *
******************************************************************************
*Br    0 :ENERGY    : ENERGY/F                                               *
*Entries :   166224 : Total  Size=     667210 bytes  File Size  =     608569 *
*Baskets :       21 : Basket Size=      32000 bytes  Compression=   1.10     *
*.............................................................................*
*Br    1 :L         : L/F                                                    *
*Entries :   166224 : Total  Size=     667085 bytes  File Size  =     594905 *
*Baskets :       21 : Basket Size=      32000 bytes  Compression=   1.12     *
*.............................................................................*
*Br    2 :B         : B/F                                                    *
*Entries :   166224 : Total  Size=     667085 bytes  File Size  =     625487 *
*Baskets :       21 : Basket Size=      32000 bytes  Compression=   1.07     *
*.............................................................................*
```

❶ 前回の LAT データを TTree にしたもの

❷ Branch が合計 23 個ある

❸ TChain を試すときに使った ROOT ファイル

❹ Branch が合計 3 個

TTree の読み書き（2）

```
$ cat src/tree_extract.C
void tree_extract(const char* input, const char* output) {
  TFile fin(input);
  TTree* photons = (TTree*)fin.Get("photons");


  Float_t energy, l, b, zenith;
  photons->SetBranchAddress("ENERGY", &energy);
  photons->SetBranchAddress("L", &l);
  photons->SetBranchAddress("B", &b);
  photons->SetBranchAddress("ZENITH_ANGLE", &zenith);

  TFile fout(output, "create");
  TTree photons_mod("photons", "");
  photons_mod.Branch("ENERGY", &energy, "ENERGY/F");
  photons_mod.Branch("L", &l, "L/F");
  photons_mod.Branch("B", &b, "B/F");

  for(int i = 0; i < photons->GetEntries(); ++i) {
    photons->GetEntry(i);
    if (zenith < 100.) {
      photons_mod.Fill();
    }
  }

  photons_mod.Write();
  fout.Close();
}
```

❶ 前回の LAT データで一部のみを抜き出したスクリプト

❷ TFile::Get を使って、名前で TTree を取り出す
※ TTree 以外も同様に取り出せる
※ キャスト（cast）という作業をする必要がある

❸ イベントごとにブランチの値を読むには、
適切な型の変数を用意しブランチに紐付ける
※ 必ず変数のポインタを渡すこと

❺ TTree::Branch を呼ぶことで、新しく作った
TTree にブランチを追加することができる
※ ここもポインタを渡す

❹ TTree::GetEntry を実行すると、指定した
ブランチのイベント毎の値が変数に代入される

❻ Fill することで、ブランチに使用している変数の
「現在」の値が詰められる

※ GetEntry する度に energy/l/b/zenith は全て
書き変わっている

型に注意

C type	ROOT typedef	C99/C++11	ROOT TTree	Python array	NumPy	FITS
signed char	`Char_t`	`int8_t`	`B`	`b`	`int8`	`A or S`
unsigned char	`UChar_t`	`uint8_t`	`b`	`B`	`uint8`	`B`
signed short	`Short_t`	`int16_t`	`S`	`h or i`	`int16`	`I`
unsigned short	`UShort_t`	`uint16_t`	`s`	`H or I`	`uint16`	`U`
signed int (32 bit)	`Int_t`	`int32_t`	`I`	`l`	`int32`	`J`
unsigned int (32 bit)	`UInt_t`	`uint32_t`	`i`	`L`	`uint32`	`V`
signed int (64 bit)	`Long64_t`	`int64_t`	`L`	`N/A`	`int64`	`K`
unsigned int (64 bit)	`ULong64_t`	`uint64_t`	`l`	`N/A`	`uint64`	`N/A`
float	`Float_t`	`float`	`F`	`f`	`float32`	`E`
double	`Double_t`	`double`	`D`	`d`	`float64`	`D`
bool	`Bool_t`	`bool`	`O`	`N/A`	`bool_`	`X`

- TTree::Branch を呼ぶときは第 3 引数で型を ROOT に教える必要がある
- C++ はポインタでメモリのアドレスが渡されるだけだと、その型を保存するのに必要なメモリの大きさが分からない

Python の場合（やり方はいくつかあります）

```
$ cat src/tree_extract.py
#!/usr/bin/env python
import ROOT
import numpy

def tree_extract(input_name, output_name):
    fin = ROOT.TFile(input_name)
    photons = fin.Get('photons')

    energy = numpy.ndarray(1, dtype = 'float32')
    l = numpy.ndarray(1, dtype = 'float32')
    b = numpy.ndarray(1, dtype = 'float32')

    fout = ROOT.TFile(output_name, 'create')
    photons_mod = ROOT.TTree('photons', '')
    photons_mod.Branch('ENERGY', energy, 'ENERGY/F')
    photons_mod.Branch('L', l, 'L/F')
    photons_mod.Branch('B', b, 'B/F')

    for i in xrange(photons.GetEntries()):
        photons.GetEntry(i)
        energy[0] = photons.ENERGY
        l[0] = photons.L
        b[0] = photons.B
        zenith = photons.ZENITH_ANGLE
        if zenith < 100.:
            photons_mod.Fill()

    photons_mod.Write()
    fout.Close()
```

❶ tree_extract.C を Python にしたもの

❷ numpy を使うやり方にします

❸ Python だと面倒な cast が不要
※ 些細なことだが慣れると C++ に戻れなくなる

❹ Python 上では直接的に C のポインタを渡せないので
numpy の ndarray を使う

❺ ここは C++ と同様、ただし引数は numpy.ndarray
※ PyROOT がうまいこと変換してくれる

❻ TTree::SetBranchAddress 不要
直接ブランチを触れる

クラスを詰める – より ROOT らしい例

```
$ root
root [0] .x event_class_tree.C+("../misc/lat_photon_weekly_w009_p302_v001.root",
"event.root")
Info in <TMacOSXSystem::ACLiC>: creating shared library /Users/oxon/git/RHEA/
src/./event_class_tree_C.so
root [1] TFile f("event.root")
(TFile &) Name: event.root Title:
root [3] photons->Print()
******************************************************************************
*Tree    :photons   :                                                        *
*Entries :   177778 : Total =         6446728 bytes  File  Size =    3336680 *
*       :          : Tree compression factor =   1.93                       *
******************************************************************************
*Br    0 :event     : PhotonEvent                                            *
*Entries :   177778 : Total  Size=    6446347 bytes  File Size  =    3332478 *
*Baskets :      447 : Basket Size=      16000 bytes  Compression=   1.93     *
*.............................................................................*
root [4] photons->Draw("event.fEnergy")
root [6] photons->Draw("event.fB:-(event.fL > 180 ? event.fL - 360 :
event.fL)>>hGal", "", "colz")
(Long64_t) 177778
```

クラスの詰め方

```
#include "TTree.h"
#include "TFile.h"

class PhotonEvent : public TObject {
private:
  Float_t fEnergy;
  Float_t fL;
  Float_t fB;
  Float_t fZenithAngle;
  Short_t fCalibVersion[3];

public:
  void SetEnergy(Float_t energy) {fEnergy = energy;}
  void SetL(Float_t l) {fL = l;}
  void SetB(Float_t b) {fB = b;}
  void SetZenithAngle(Float_t zenith) {fZenithAngle = zenith;}
  void SetCalibVersion(Short_t* calib) {
    for(int i = 0; i < 3; ++i) {
      fCalibVersion[i] = calib[i];
    }
  }

  ClassDef(PhotonEvent, 1)
};

void event_class_tree(const char* input, const char* output) {
  TFile fin(input);
  TTree* photons = (TTree*)fin.Get("photons");
  Float_t energy, l, b, zenith;
  Short_t calib[3];
  photons->SetBranchAddress("ENERGY", &energy);
  photons->SetBranchAddress("L", &l);
  photons->SetBranchAddress("B", &b);
  photons->SetBranchAddress("ZENITH_ANGLE", &zenith);
  photons->SetBranchAddress("CALIB_VERSION", calib);

  PhotonEvent event;

  TFile fout(output, "create");
  TTree photons_mod("photons", "");
  photons_mod.Branch("event", "PhotonEvent", &event, 16000, 0);

  for(int i = 0; i < photons->GetEntries(); ++i) {
    photons->GetEntry(i);
    event.SetEnergy(energy);
    event.SetL(l);
    event.SetB(b);
    event.SetZenithAngle(zenith);
    event.SetCalibVersion(calib);
    photons_mod.Fill();
  }

  photons_mod.Write();
  fout.Close();
}
```

❶ コンパイルするので必要なものを #include

❷ クラスを作る。TObject から継承しなくても良い。

❸ メンバ変数の型は、メモリサイズの環境依存を
減らすために ROOT で typedef されたものを使う

❹ メンバ変数を private にする場合は setter を
本当は getter も必要だけど、この例では使わない

❺ ROOT で class を追加するときのおまじない

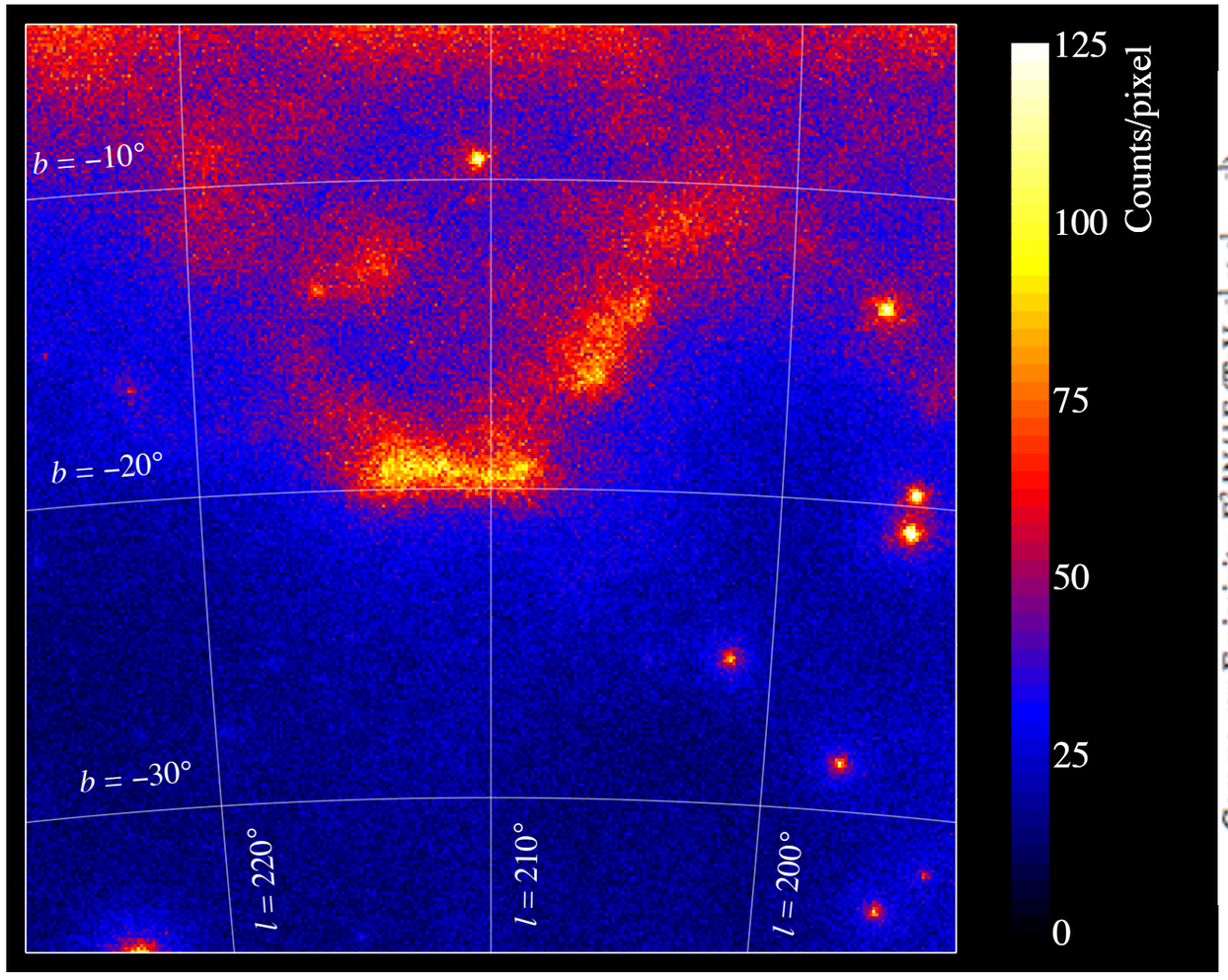
❻ クラスのインスタンスのポインタを渡す

❼ クラスのメンバ変数を更新して詰めるだけ

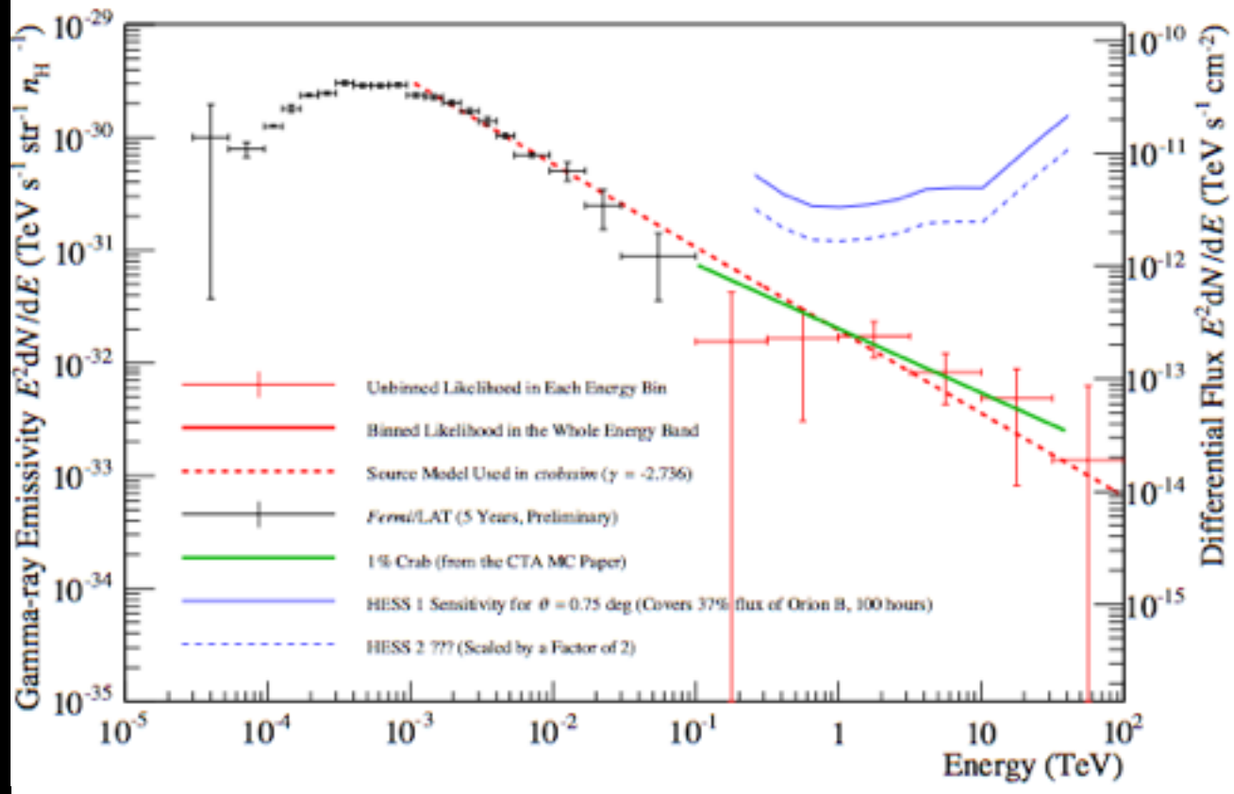
雑多な話

どんな風に ROOT を普段使っているのか (1)

Fermi/LAT のカウントマップの例

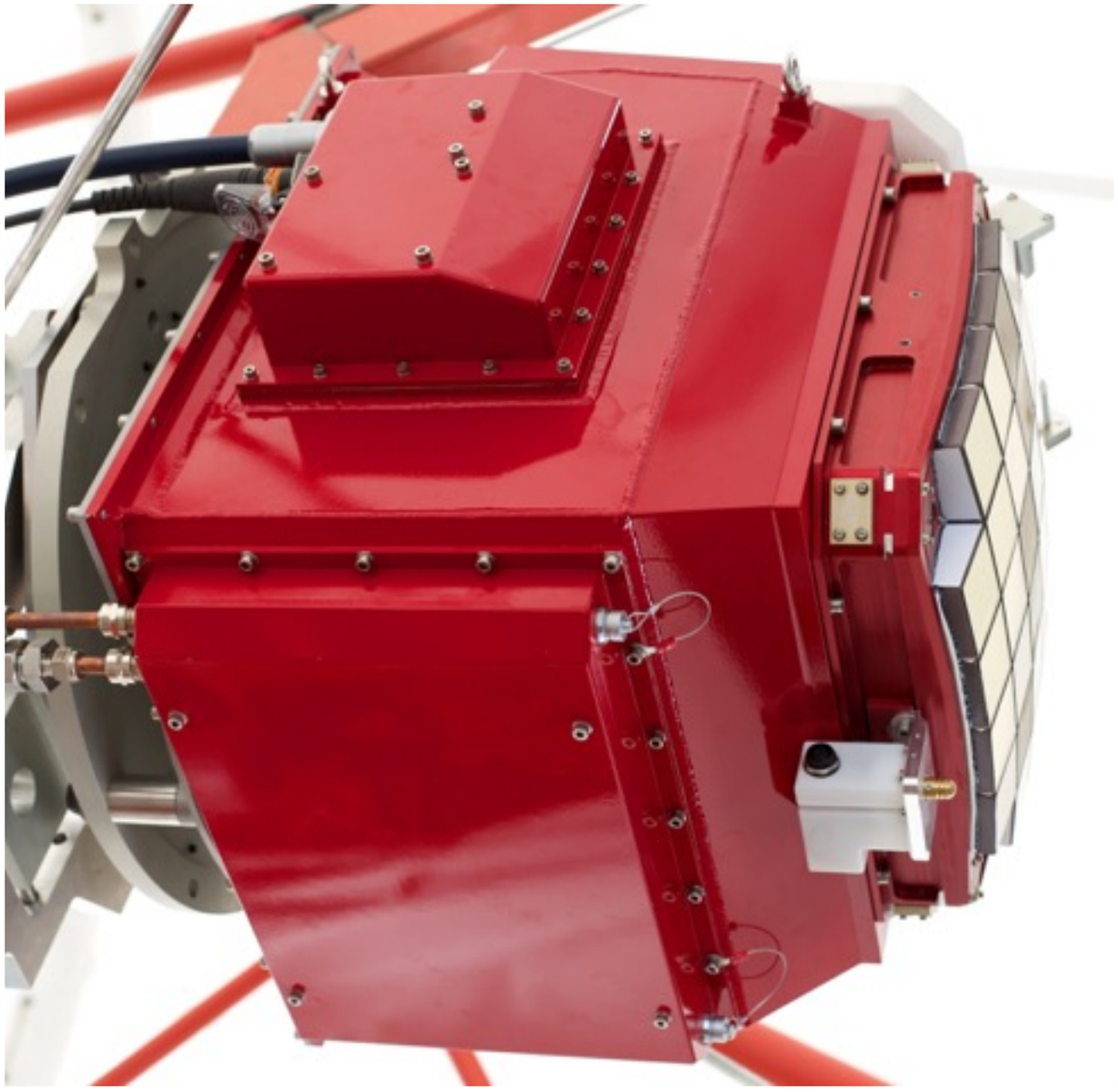


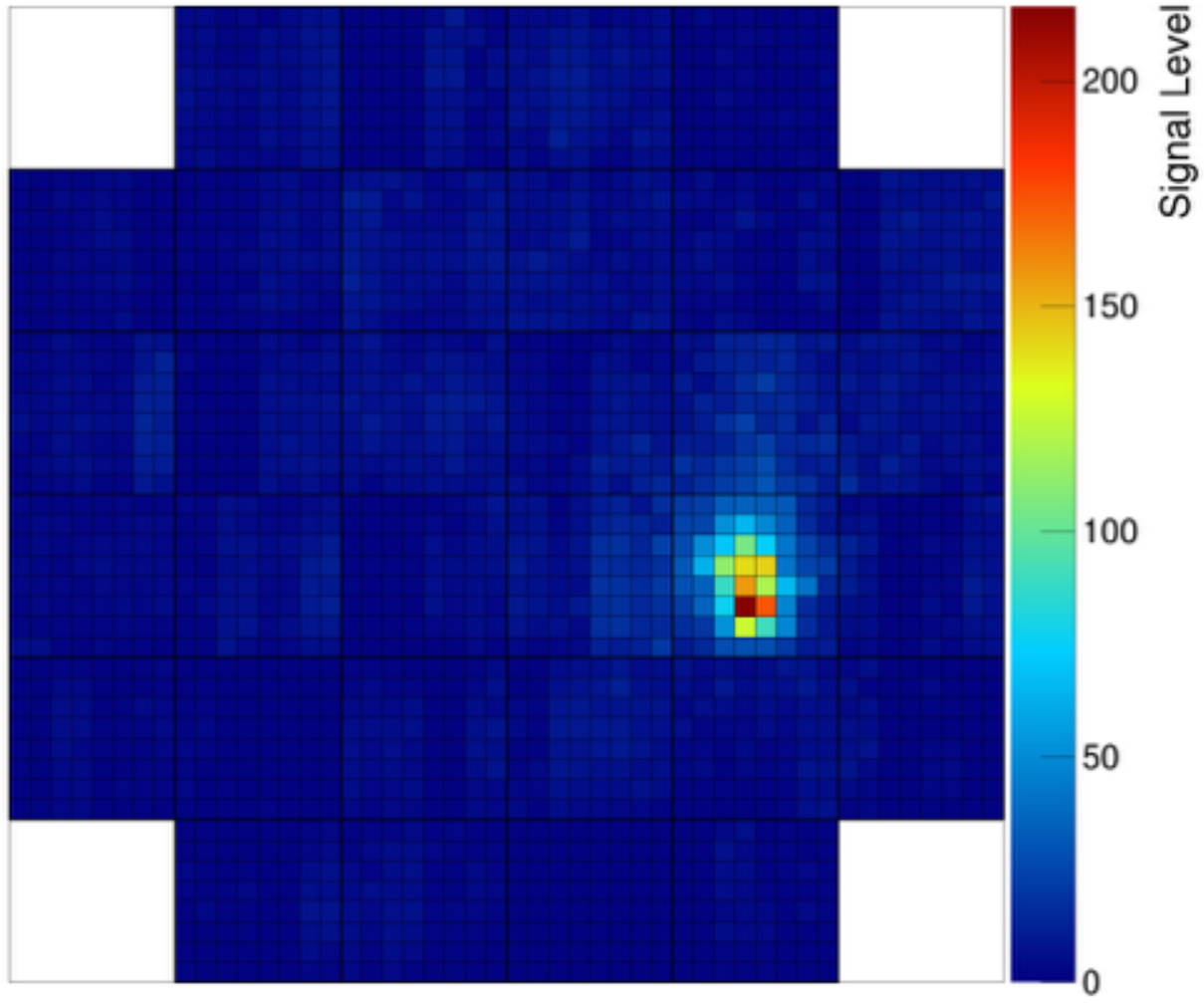
CTA のシミュレーションの例



- ガンマ線観測のデータ解析とシミュレーション（の結果の表示）
- 計算自体は Fermi や CTA で ROOT に依存しないソフトがやってくれる
- 出てきたガンマ線スペクトルのフィット、カウントマップの表示など

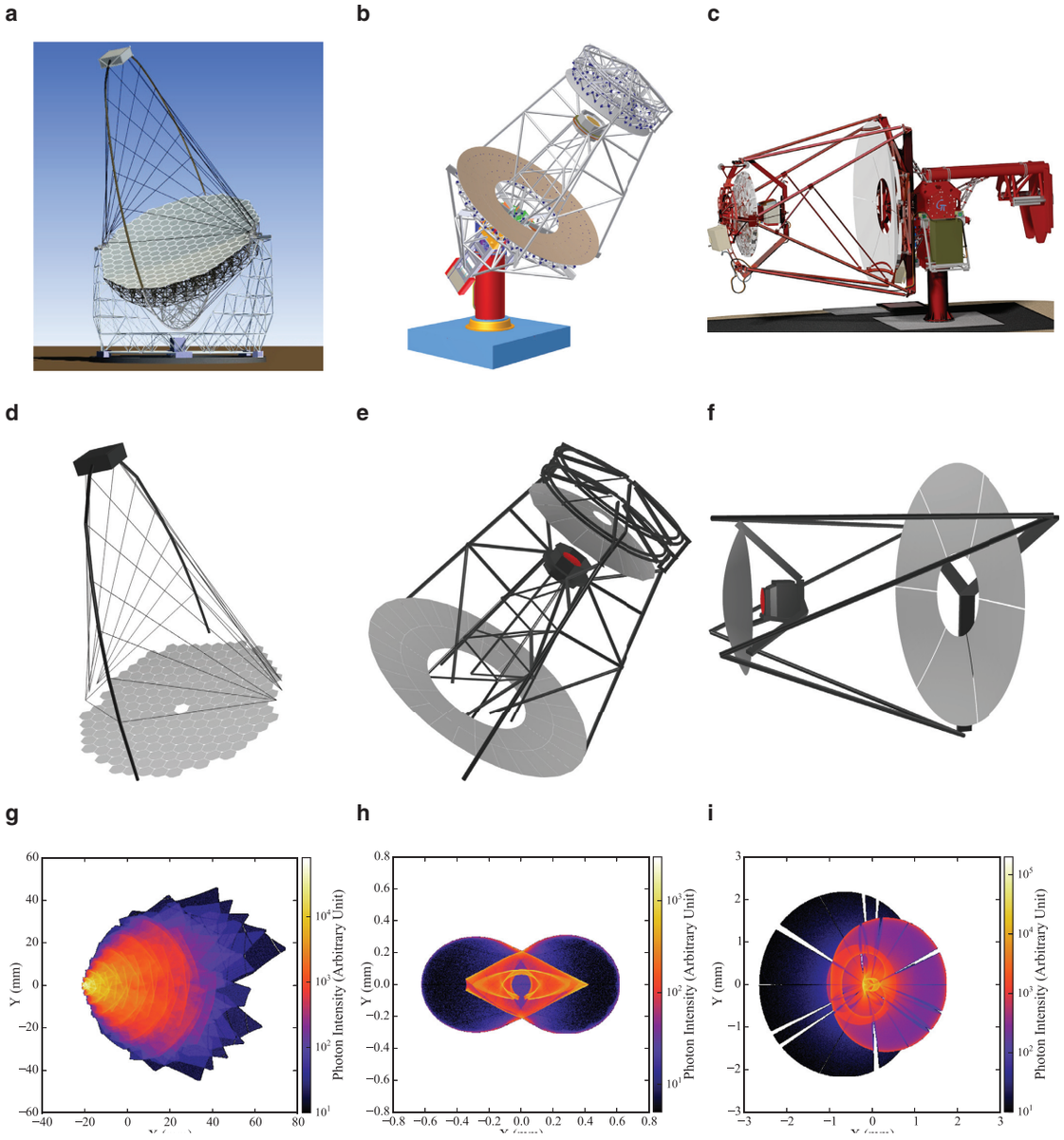
どんな風に ROOT を普段使っているのか (2)





- CTA の望遠鏡カメラ試作機で取得したデータの解析
- エレキの性能試験
- DAQ 部分には ROOT は使っていない

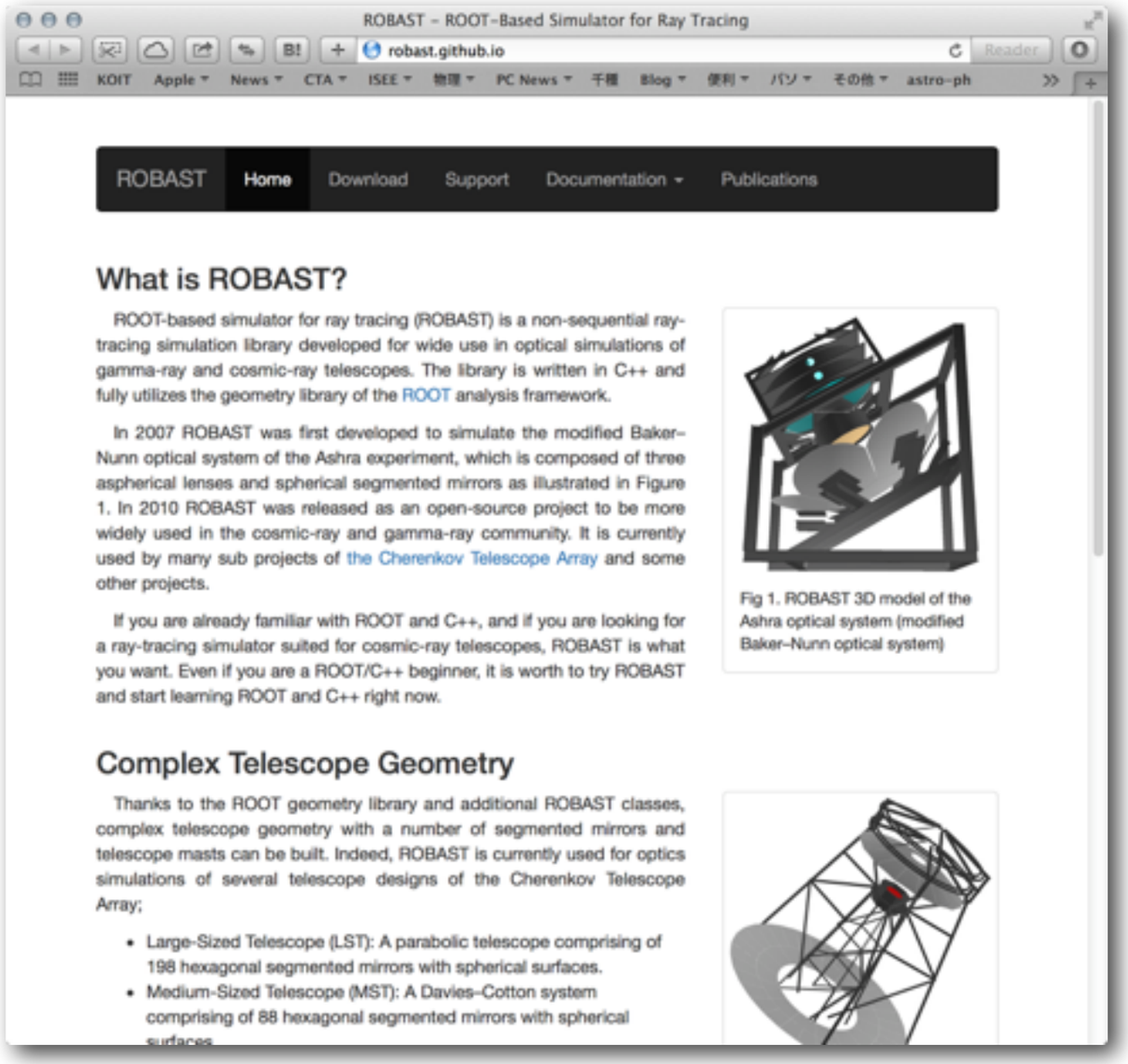
どんな風に ROOT を普段使っているのか（3）



- CTA の光学系シミュレーション
- 6 種類ある光学系のうち 4 種類で ROBAST（後述）が使用されている
- 光学系の性能評価を、光線追跡結果を TH2 に詰めて解析することで行う

ROBAST

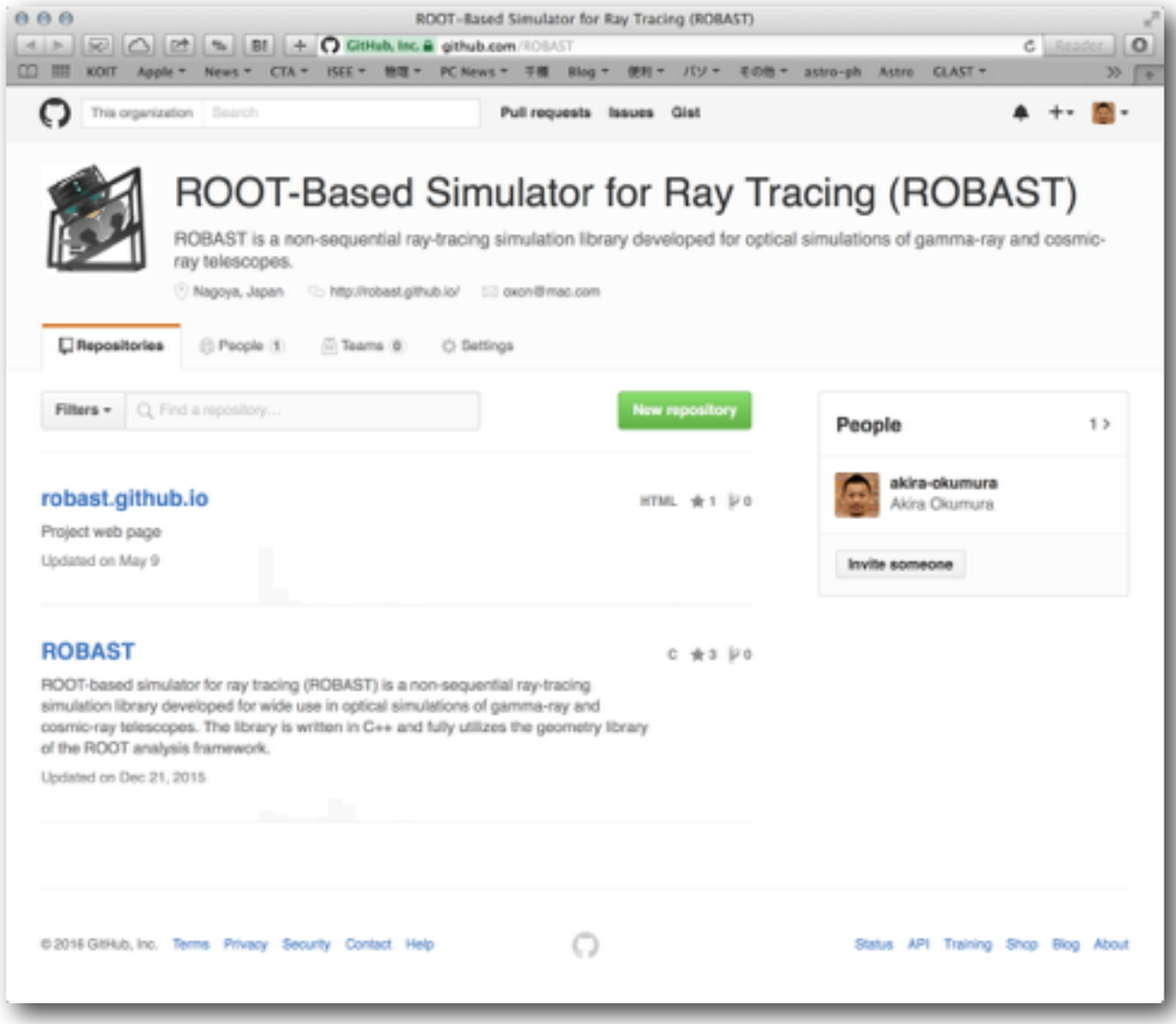
http://robast.github.io/



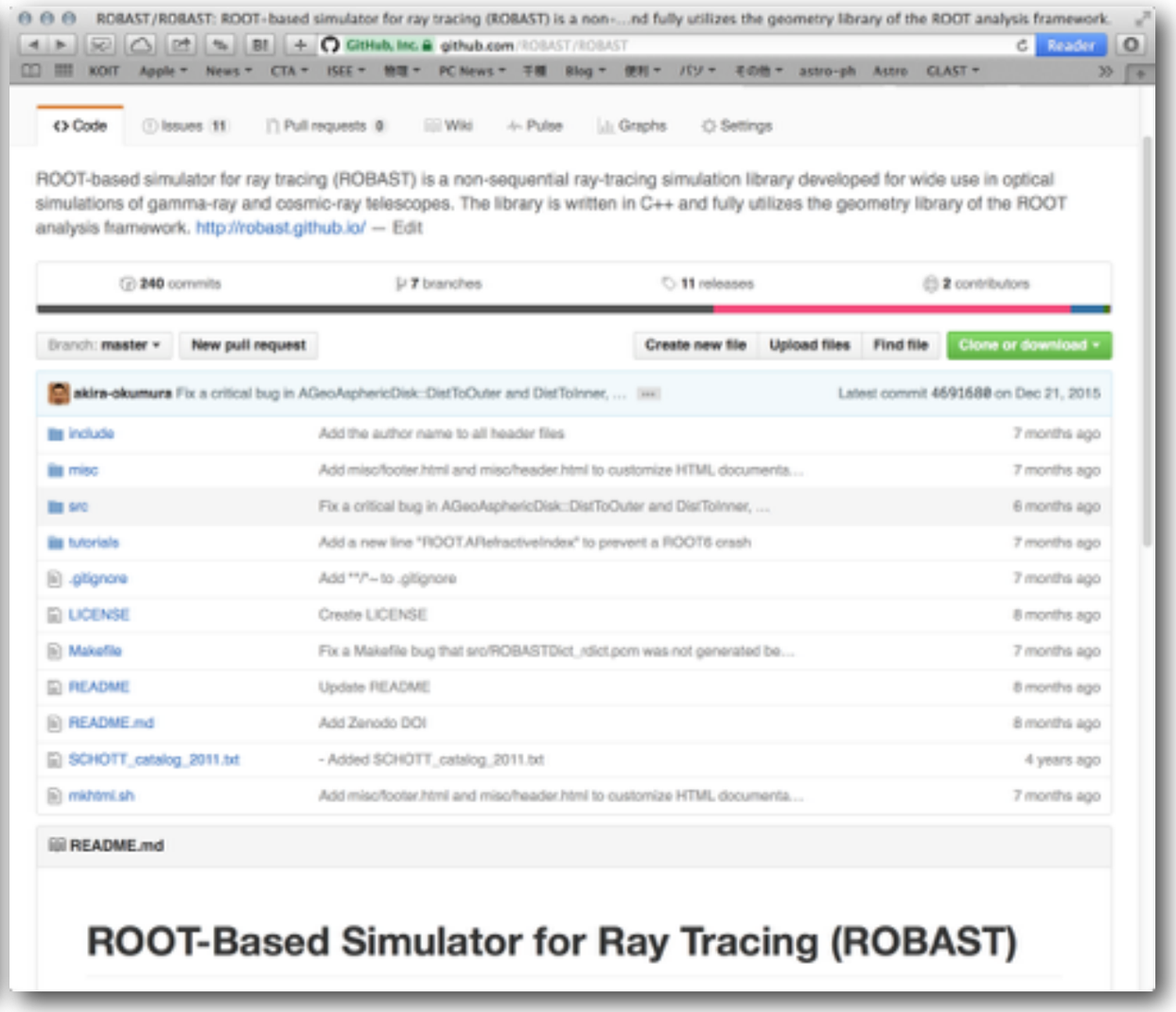
- 宇宙線実験屋向けの光線追跡ライブラリ
- ROOT が持っている機能を多数利用（多分、自分で書いた部分は 3000 行くらいしかない）
- 比較的小規模なので、ROOT を利用したライブラリの作り方の参考になるかも

ROBAST の GitHub レポジトリ

https://github.com/ROBAST



https://github.com/ROBAST/ROBAST



- ROOT で自作ライブラリを作るときの、ひとつの例
- 2007 年ごろに書いたものなので、少し汚い
- GitHub での webpage の公開の仕方とかの参考にも

自分は関係していませんが、よその宣伝

wiki.kek.jp/display/PPCC/PPCC-SS-2017

Confluence Spaces Create Log in

Pages / Particle Physics Computing Consortium / Event

PPCC-SS-2017

Created by NAKAMURA Tomoaki, last modified on Jun 06, 2017

第一回粒子物理コンピューティングサマースクール（PPCC-SS-2017）

開催期間・会場
2017年7月31日（月） ～ 8月4日（金）
KEKつくばキャンパス
- 3号館1階会議室（プレナリ）
- 1号館1階談話室1/2（パラレル）

参加申し込みについて
対象: 修士課程学生
定員: 35名（定員になり次第、締め切ります）
参加申し込みについての詳細

事前準備について
講習・実習は参加者に配布する仮想マシーンを使います。64 bit OS（Windows10, Mac OS X, Linux）上にハイパーバイザ（Oracle VM VirtualBox）をインストールしたノートPCを持参してください。
事前準備についての詳細

謝辞
第一回粒子物理コンピューティングサマースクールは、東京大学素粒子物理国際研究センターと高エネルギー加速器研究機構による平成29年度大学等連携支援事業からのサポートを受けています。

お問い合わせ
粒子物理コンピューティング懇談会事務局
E-mail: ppcc-sec@ml.post.kek.jp

開催趣旨
粒子物理コンピューティング懇談会（PPCC）は昨年より活動を開始した、素粒子・原子核・宇宙物理関連のコンピューティング技術利用に関するコミュニティです。その活動の一端として7月31日～8月4日の5日間、高エネルギー加速器研究機構を会場に「第一回コンピューティングサマースクール」を開催します。

今日、粒子物理（素粒子・原子核・宇宙物理）分野では大規模な検出器を用い大量のデータを収集することから、コンピューティングへの要請が非常に高まっています。新しい実験を始めるためには最先端のコンピューティング・ソフトウエア技術の導入が欠かせません。それを担う研究者の育成が急務であることは間違いありません。

そういった教育環境が整った研究機関は日本にはそう多くありません。そこで、全国のボランティアの協力を得て、コンピューティングについて集中的なトレーニングコースを設けることにしました。アナリシス・デザインプロセスから始めるオブジェクト指向プログラミング技術、多変量解析や機械学習、検出器シミュレーションなどの先端ソフトウエア、グリッドやクラウドなど分散コンピューティング技術などを学習します。

サマースクールでは最新技術の講義を行うとともに、各人が課題を決め4日間かけてプログラム開発を行う実習を予定しています。対象として修士課程の大学院生を想定していますが、勉強してみたいという方はどなたも大歓迎です。参加をお待ちしています。

校長　坂本　宏

プログラム

時限	7/31(月)	8/1(火)	8/2(水)	8/3(木)	8/4(金)
9:00 – 10:30	1) 開会あいさつ 2) 参加者案内 3) 実習テーマ説明	6) 解析フレームワーク (Root, RooFit, RooStats)	9) ネットワークの仕組み	パラレル a), b), c)	発表会
11:00 – 12:30	4) プログラミング言語 (C++)	7) 多変量解析 (TMVA)	10) CentOS 7の仕組み	パラレル a), b), c)	発表会
12:30 – 13:30	昼休憩	昼休憩	昼休憩	昼休憩	施設見学
13:30 – 15:00	5) プログラミング言語 (Python)	8) 研究ツール (Jupyter Notebook)	11) Grid概要	パラレル a), b), c)	解散
15:30 – 18:00	実習	実習	実習	発表資料作成	

共通講習担当講師（月・火・水）

これから

- ともかく実験やデータ解析を頑張る
  - ROOT や C++/Python の勉強だけしても意味がない
  - 必要になれば、自ずと勉強するべき機能が分かってくる
- 統計学の基本
  - 誤差とは何か、確率分布とは何かを改めて学ぶ
  - フィッティングを実データでやってみる
- 発展的な ROOT や C++/Python の学習
- 教員、先輩にたくさん質問をする
- 来年はぜひ講習会のサポート役に回ってください