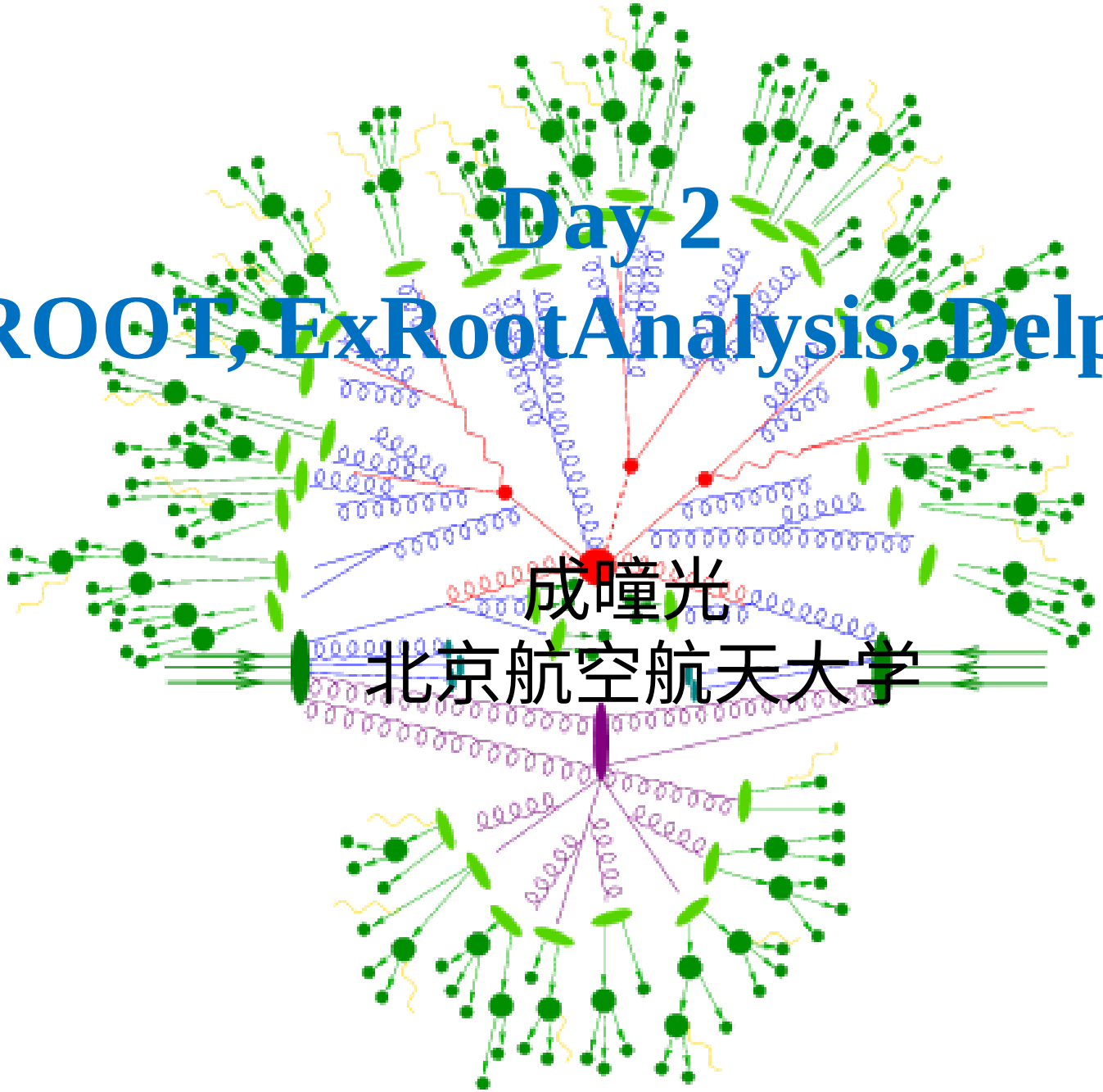
Day 2
ROOT, ExRootAnalysis, Delphes



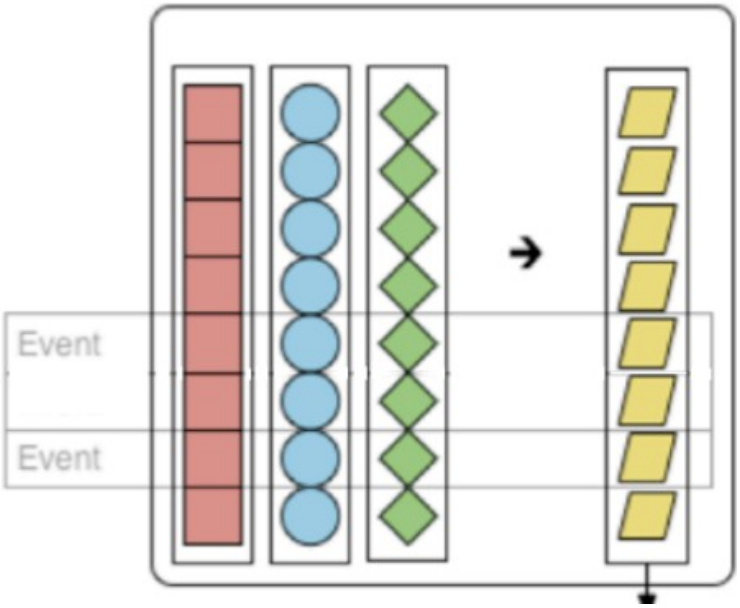
成曈光
北京航空航天大学

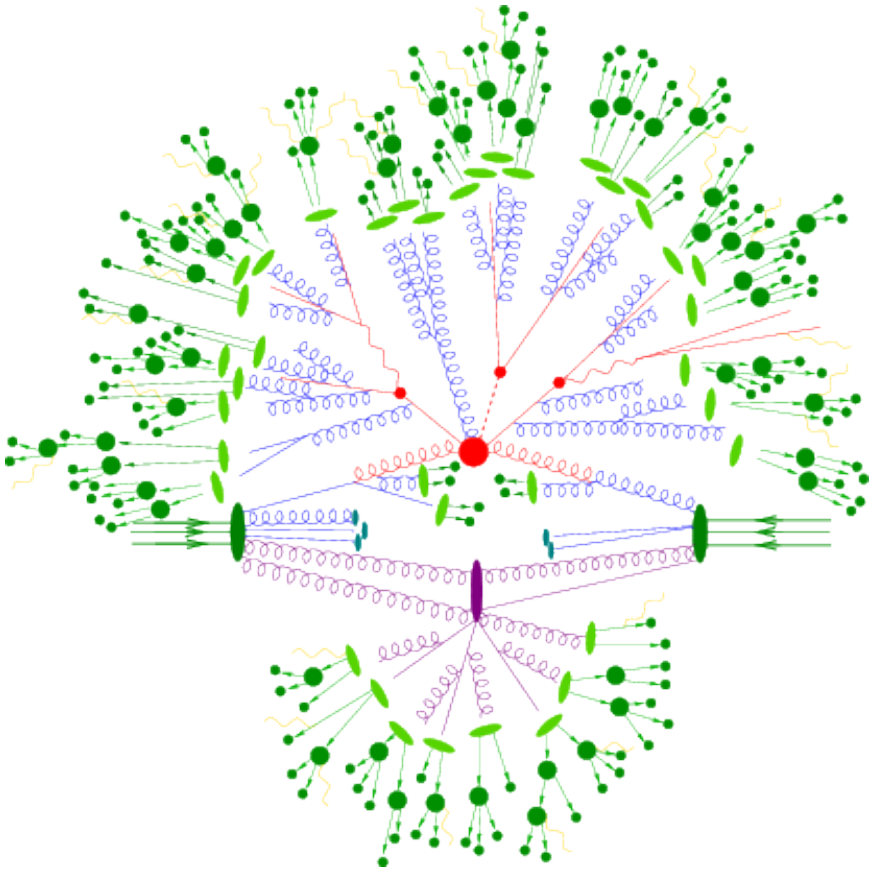
Reference

- 第一届中国 CMS 冬令营
  https://indico.ihep.ac.cn/event/15418/
- ROOT online resources
  - manual : https://root.cern/manual/basics/
  - tutorial:
    https://root.cern.ch/doc/master/group__tutorial__tree.html
  - etc..

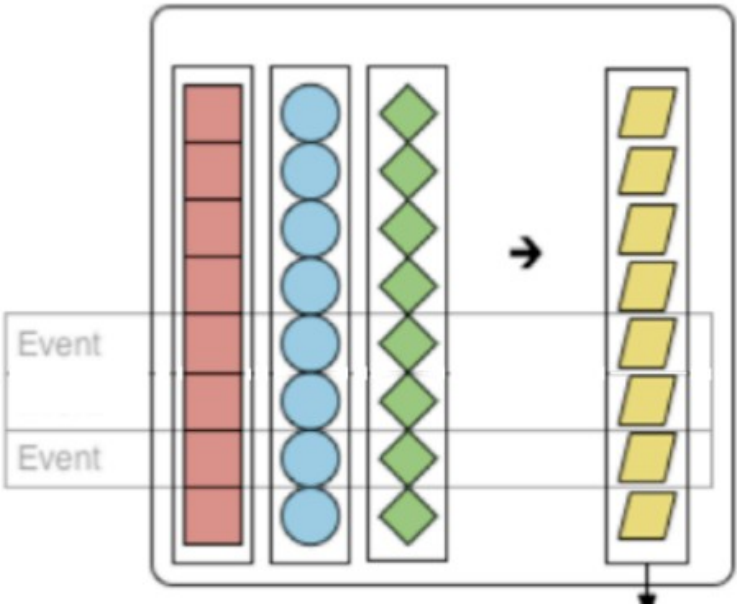
pp collision data : LHC and HEPMC level

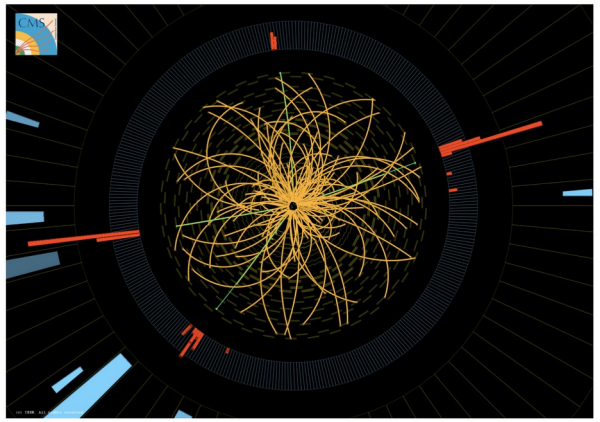
Columnar data

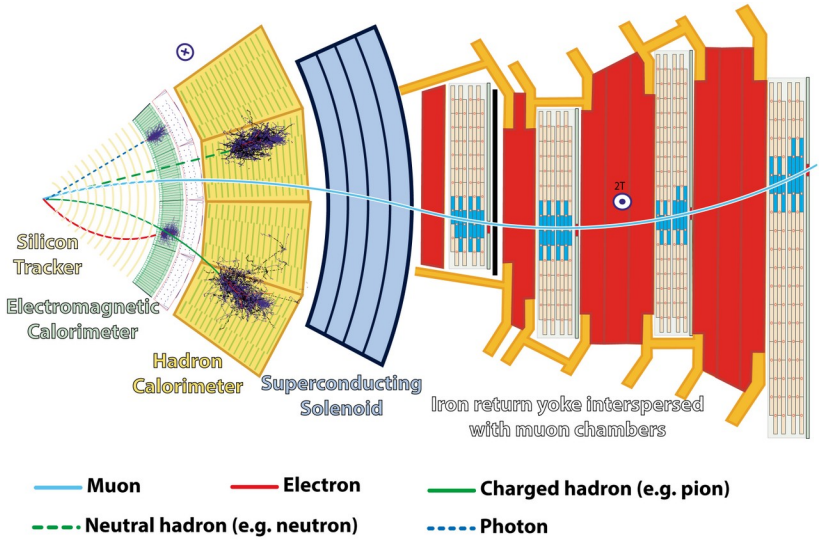




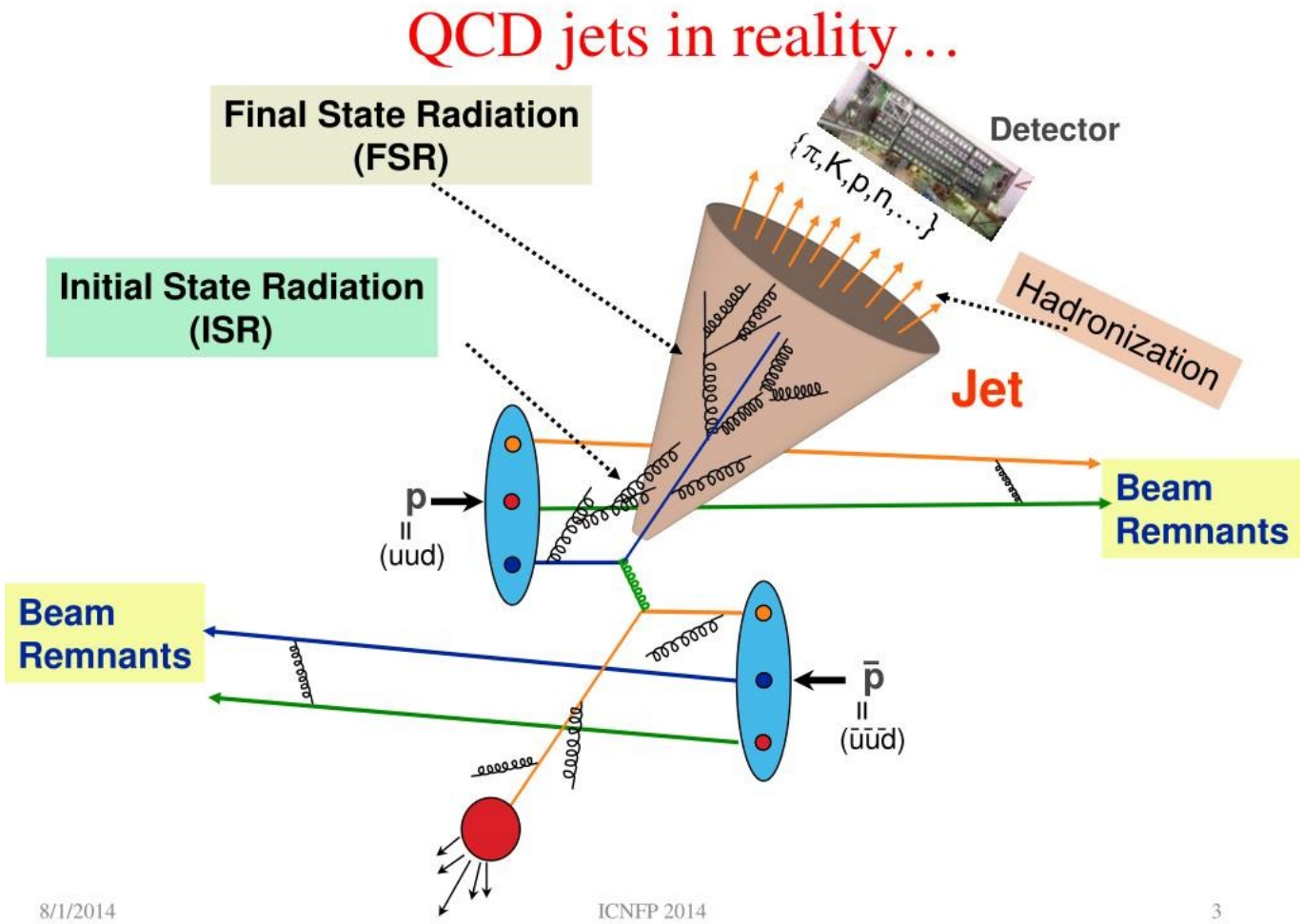
pp collision data : reconstruction (Delphes level)



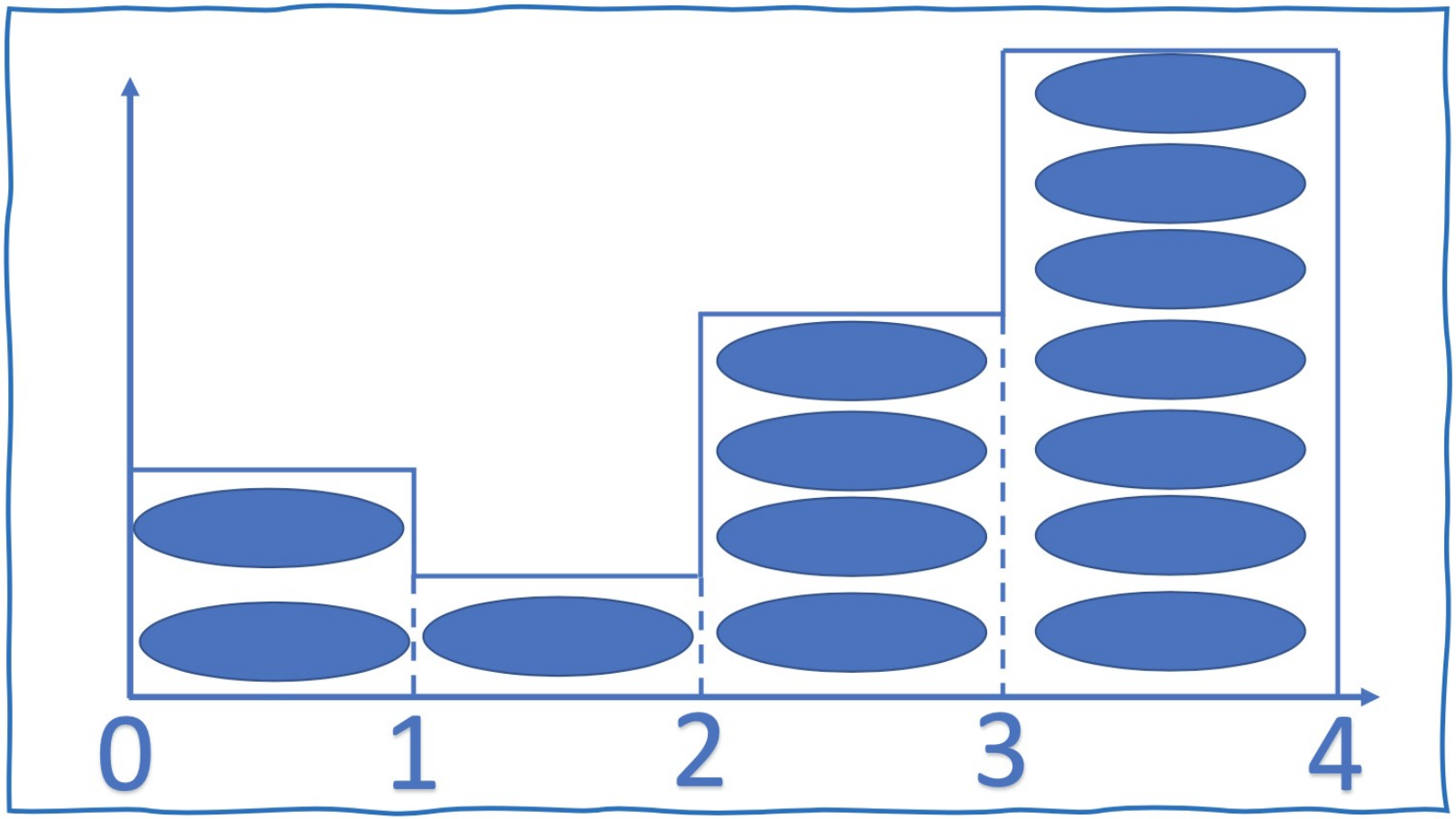




pp collision data : jet

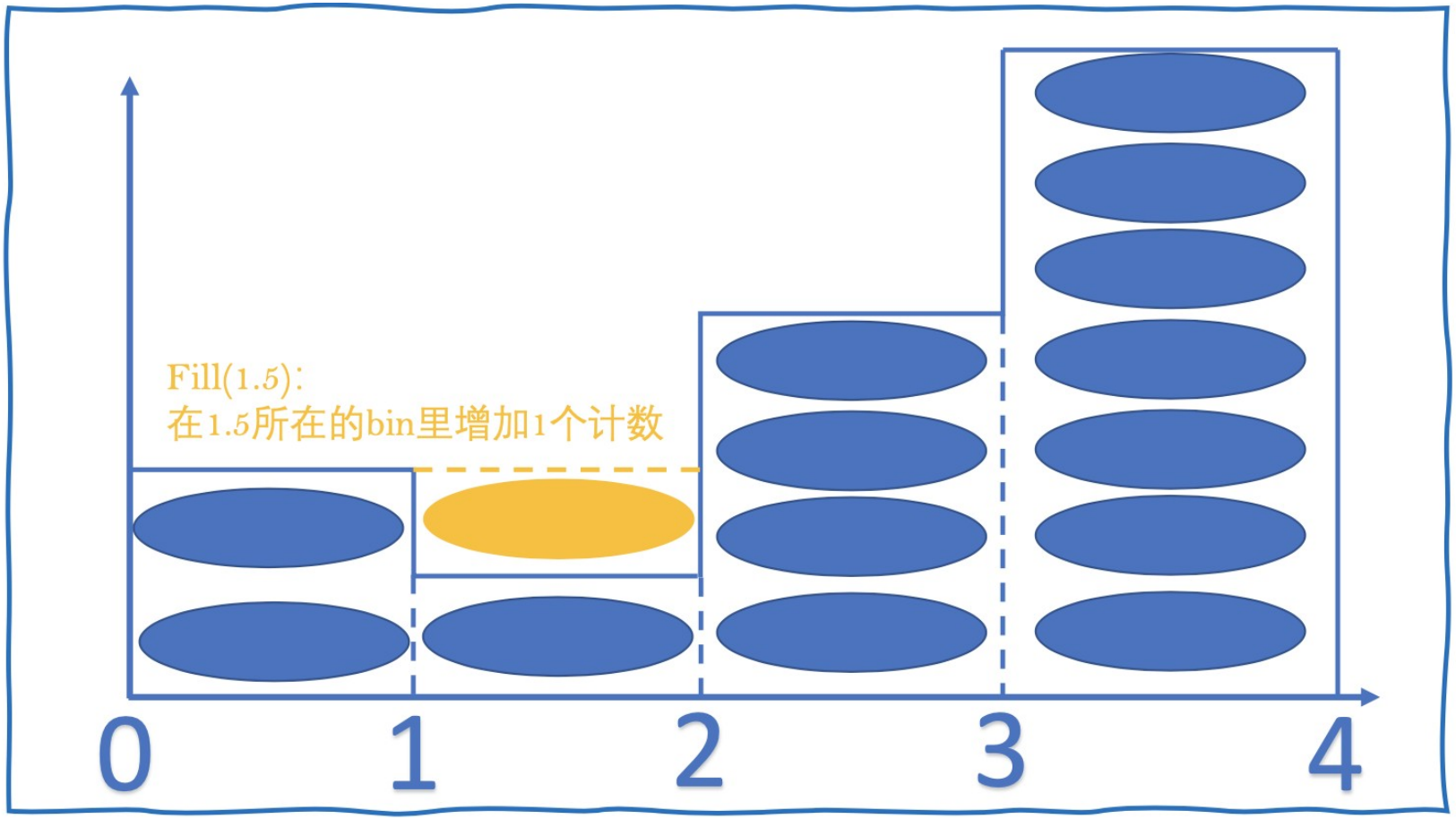


Two types of ROOT data container for “event” data : histogram (binned)



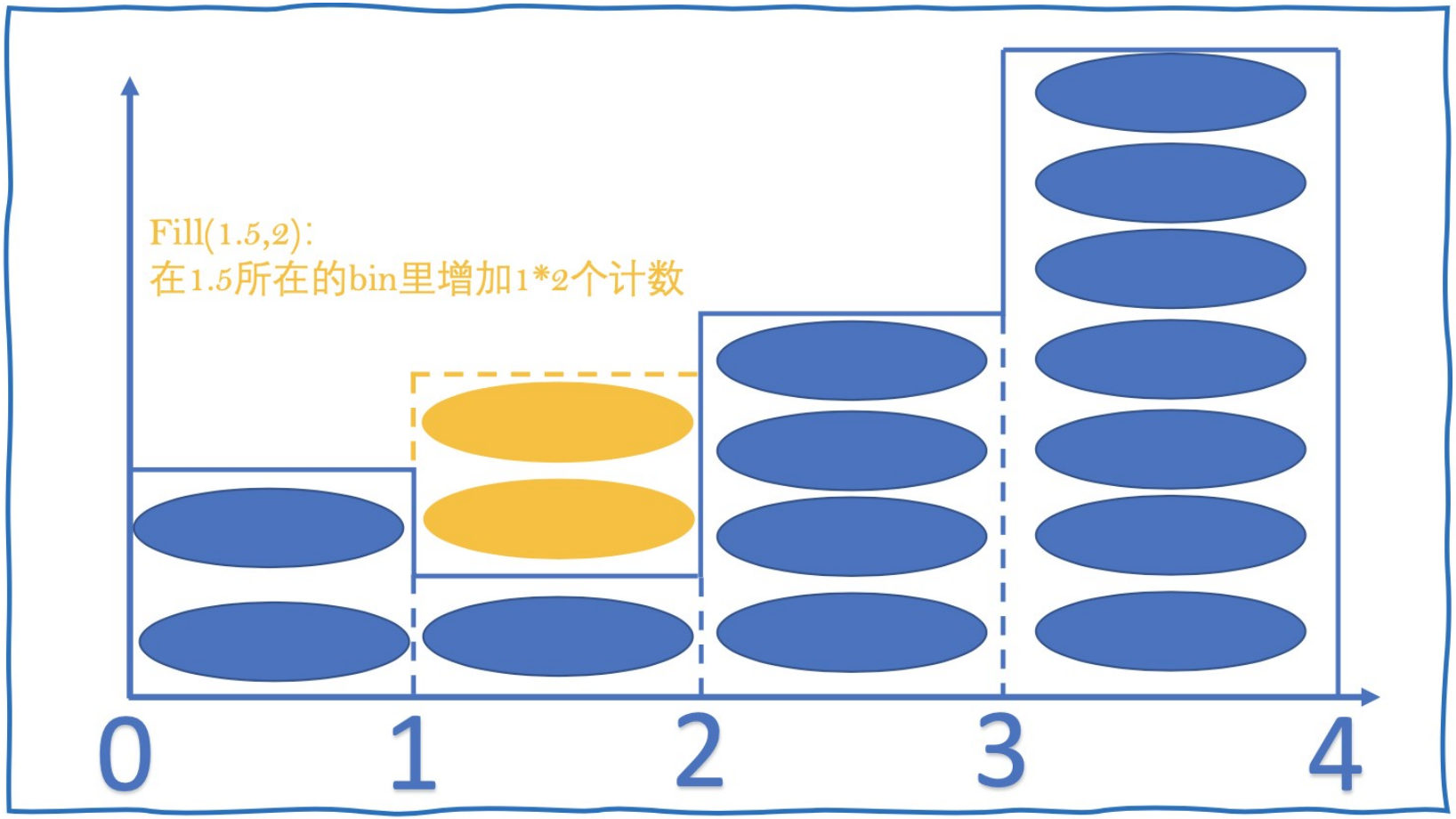
bin ： 变量分立话，比如 pp > e+e- 中的 e+e- 不变质量

Two types of ROOT data container for “event” data : histogram (binned)



将某一个 event 的信息填充

Two types of ROOT data container for “event” data : histogram (binned)

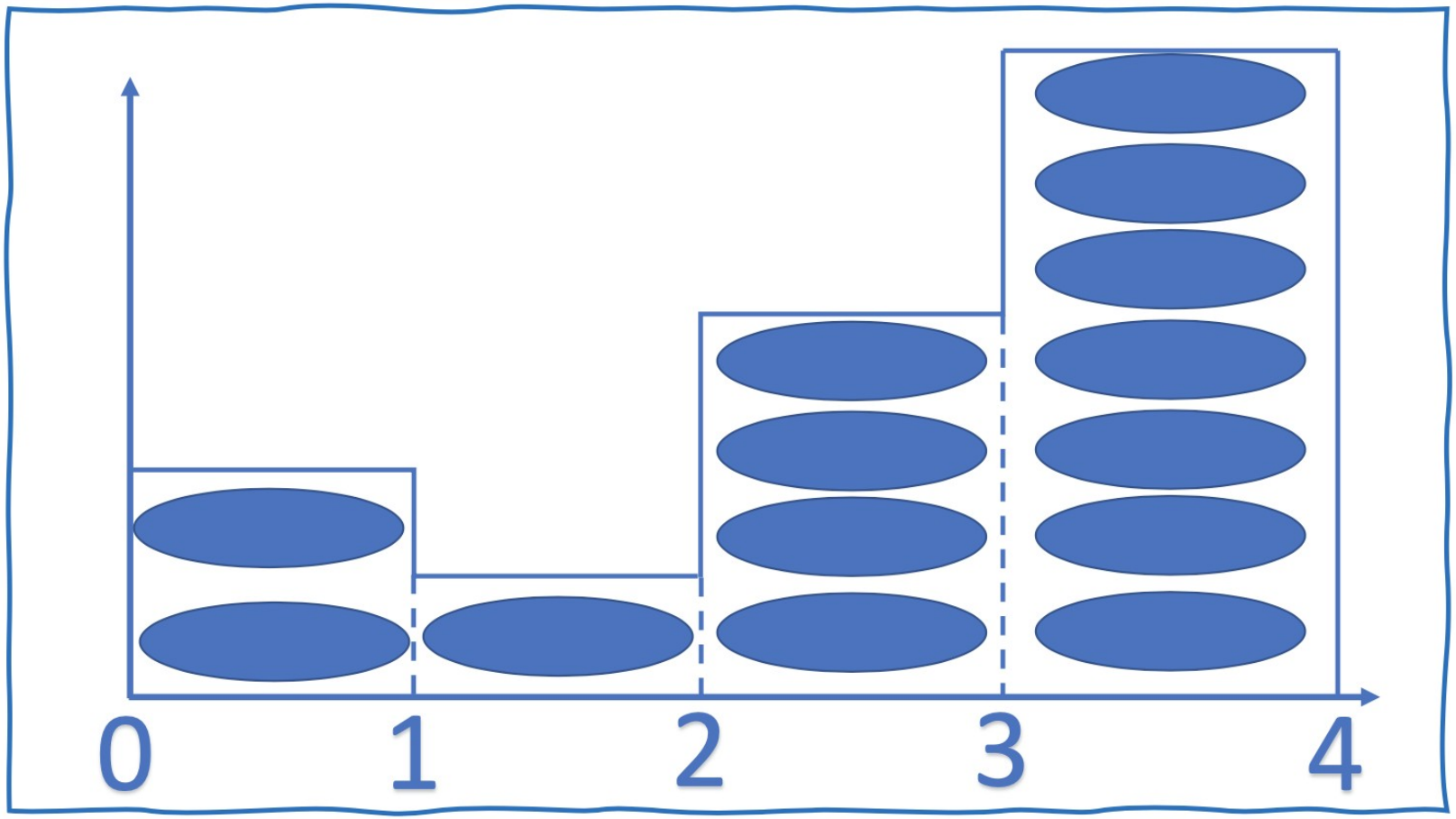


继续将令一个 event 的信息填充

Two types of ROOT data container for “event” data : histogram (binned)

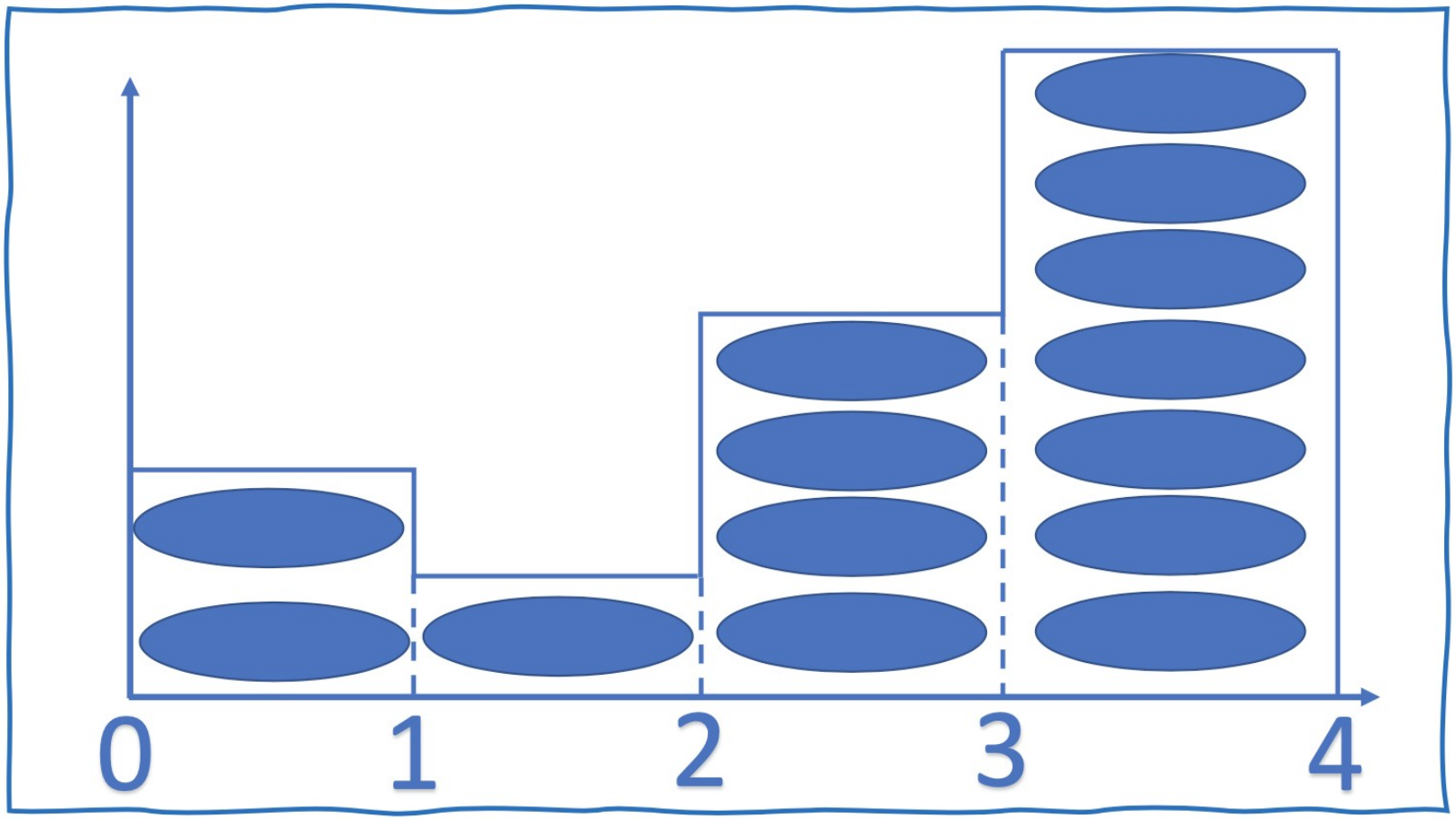
```
const int NEntry= 10000 ;
TH1F *h1 = new TH1F("h1","A simple histo",100,0,1);
// 填充直方图 10000 次，用 (0,1) 均匀分布
for (int i=0;i<NEntry;i++)
h1->Fill( gRandom->Uniform() );
```

Two types of ROOT data container for “event” data : histogram (binned)



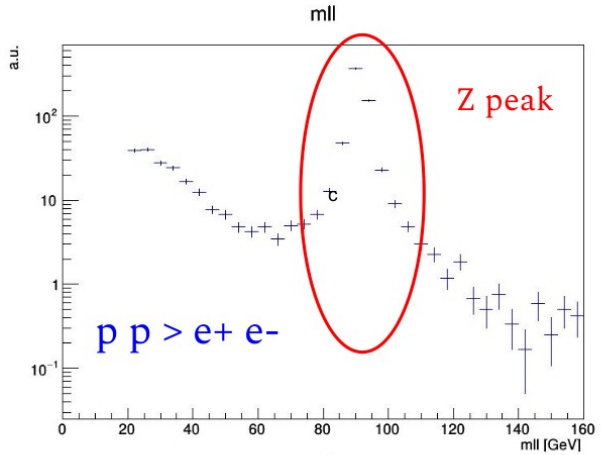
每个格子（bin）的高低代表了相应的样本变量的分布 ~PDF

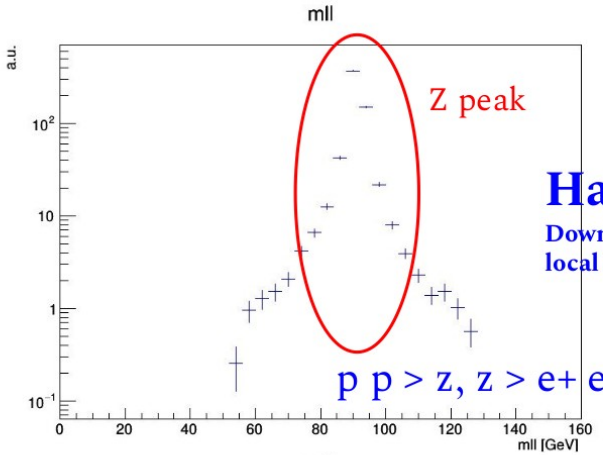
Two types of ROOT data container for “event” data : histogram (binned)



将不同样本的 events 填充结果结合起来的时候就要考虑归一化问题

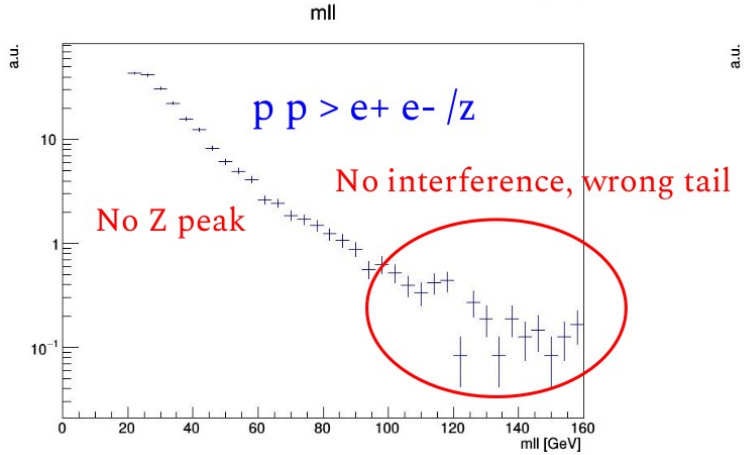
Two types of ROOT data container for “event” data : histogram (binned)

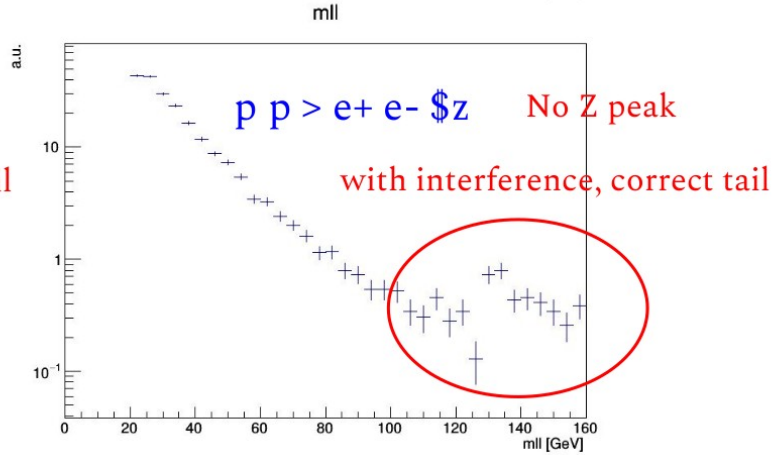




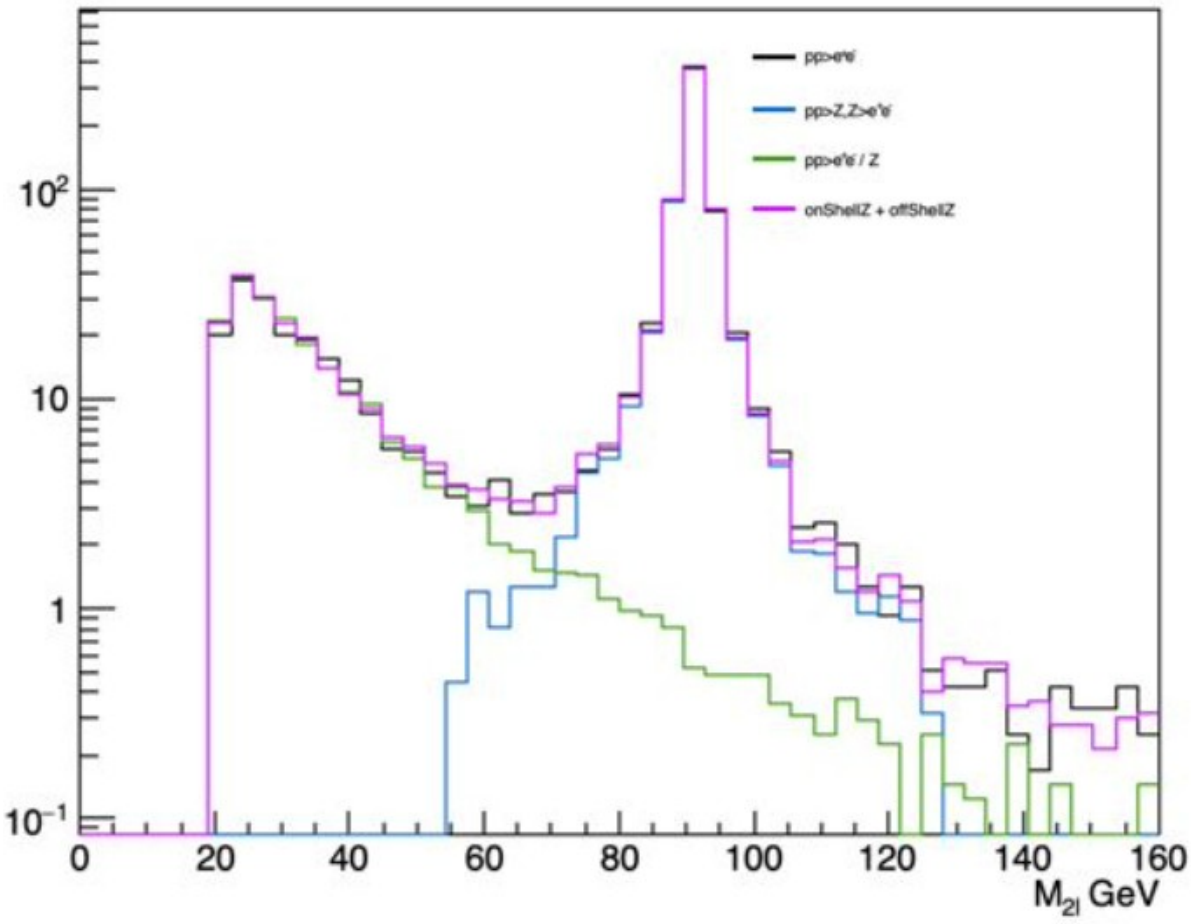
Hands-on

Download plot to your local

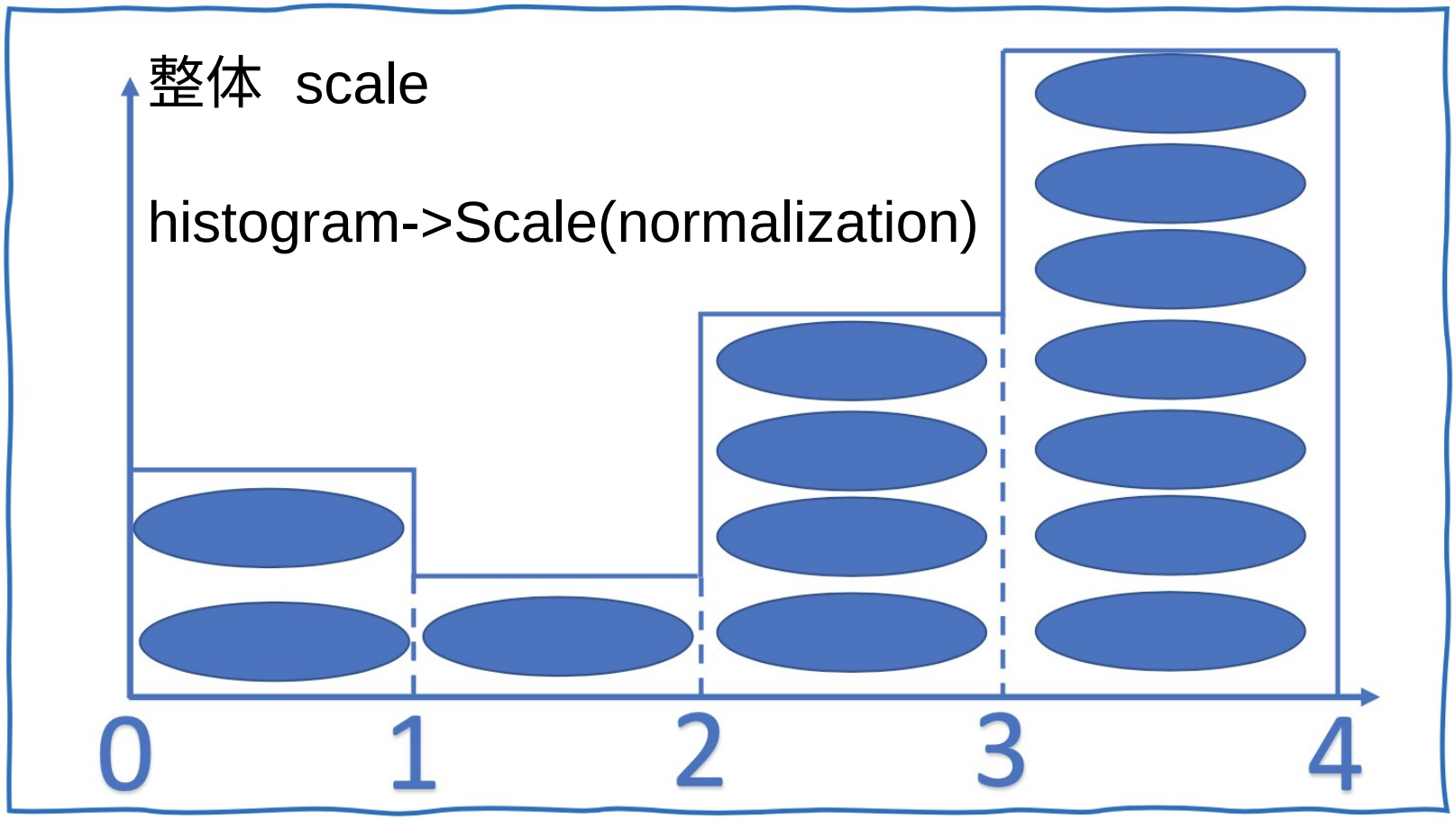




Two types of ROOT data container for “event” data : histogram (binned)



Two types of ROOT data container for “event” data : normalization



Two types of ROOT data container for "event" data : normalization

填充 weighted events

histogram->Fill(m(e+e-), weight)

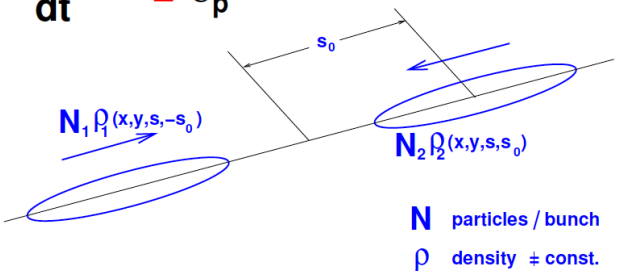
如果 weight 是 cross section，
那么 histogram 的 normalization 就是
Number filled events X cross section

如果此时想把 histogram" 归一化到 cross section"S，
则 histogram->Scale(1/ total number of events)

Number of events: $N = \sigma \cdot \int \mathcal{L} dt$

Luminosity at the LHC (pp collistion)

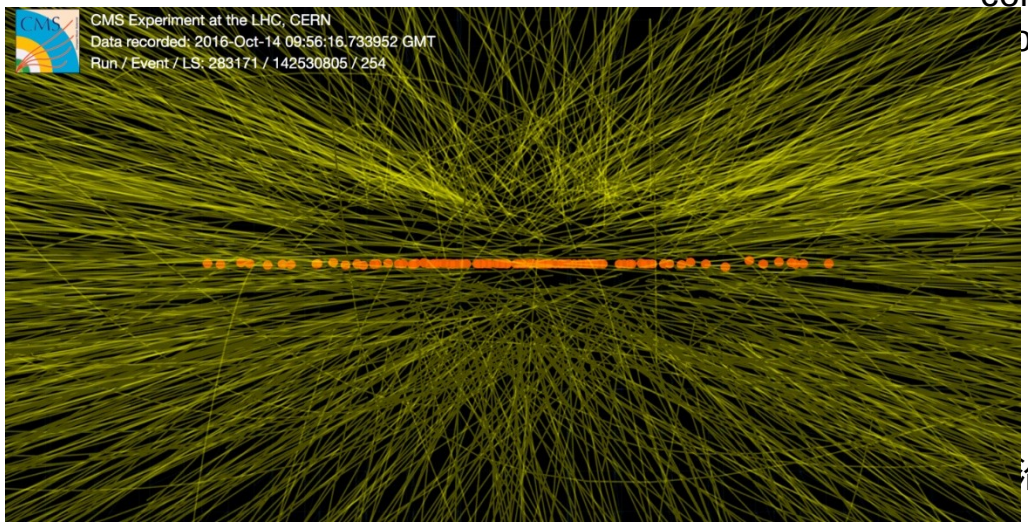
$$\frac{dR}{dt} = L\,\sigma_p$$

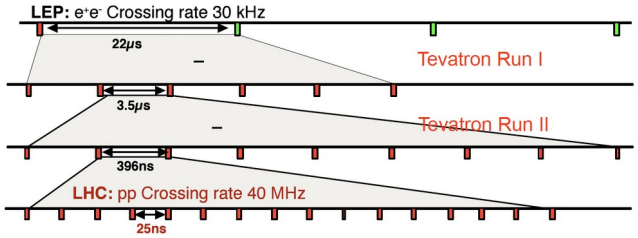


$$\mathcal{L} = \frac{N_1 N_2 f N_b}{2\pi\sqrt{\sigma_{1x}^2+\sigma_{2x}^2}\sqrt{\sigma_{2y}^2+\sigma_{2y}^2}}$$

LHC parameters

- Protons/bunch : ~10^{11}
- Bunch spacing : 25ns
- Max # of bunches : 27km/(c*25ns) ~ 3600
- Luminosity : 2x10^{34} $cm^{-2}s^{-1}$
- Average number of interactions per bunch crossing (**in-time pileup**) :
  n = L x $\sigma_{minbias}$ x 25ns x (3600/2556) ~ 50-60
  - **out-of-time pileup** :
    contribution from different(previous) bunch
    ossings

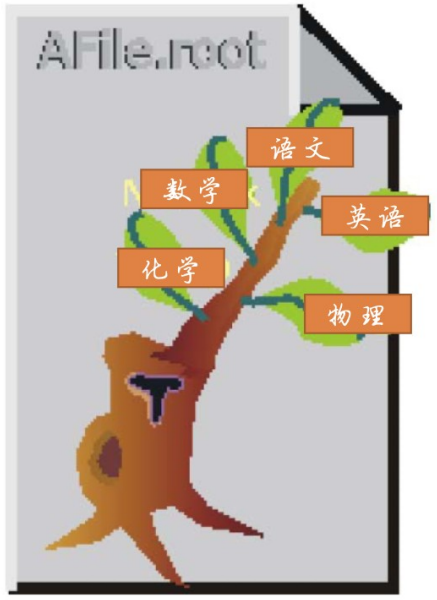




沿讲座

Two types of ROOT data container for “event” data : TTree (unbinned)

	A	B	C	D	E	F	G
1				月考成绩			
2	学生姓名	语文	数学	英语	物理	化学	生物
3	张三	103	120	130	90	90	90
	李四	90	150	110	100	100	100



Branch -> 对应列
Leaf -> 对应列中的每个元素

相比 Excel
可以存储更复杂的数据，
branch 存储的数据可以是一个复杂的数据类型
比如 std::vector, array, customized data type

Two types of ROOT data container for “event” data : TTree (unbinned)

```
TFile*f = new
TFile("tree1.root","recreate");
TTree*t1 = new
TTree("t1",”test");
float px,py,pz; float random;
int i;// 设置 Tree t1 的 branch
t1->Branch("px",&px,"px/F");
t1->Branch("py",&py,"py/F");
t1->Branch("pz",&pz,"pz/F");
t1->Branch("i",&i,"i/I");
for (i=0;i<5000;i++) {
px = gRandom->Gaus(0,1);
pz = gRandom->Gaus(0,1);
pz = gRandom->Exp(10);
random = gRandom->Rndm();
t1->Fill();// 填充 Tree t1
}
t1->Write();
```

```
const Int_t kMaxTrack= 50;
int ntrack;
float px[kMaxTrack];
float py[kMaxTrack];
TFile* f = new
TFile("tree2.root","recreate");
TTree*t2 = new
TTree("t2","Reconstevents");
t2>Branch("ntrack",&ntrack,"ntrack/
I");
t2->Branch("px",px,"px[ntrack]/F");
t2->Branch("py",py,"py[ntrack]/F");
```

Two types of ROOT data container for “event” data : TTree (unbinned)

```
TFile*f = new TFile("tree2.root","read");
TTree*t2 = (TTree*)f->Get("t2"); // 步骤 1: 定义好必要的变量
const Int_t kMaxTrack= 100; Int_t ntrack;
Float_t px[kMaxTrack]; //[ntrack]
Float_t py[kMaxTrack]; //[ntrack]

// 步骤 2: 用 SetBranchAddress 函数
// 将 tree 的 Branch 与定义好的变量
// 地址联系起来。
t2->SetBranchAddress("ntrack", &ntrack);
t2->SetBranchAddress("px", px);
t2->SetBranchAddress("py", py);

// 步骤 3: 对所有事例循环
for (int i=0;i<nentries;i++) {
 t2->GetEntries(i)
 int Ntrack = ntrack;
 for(int j = 0, j<ntrack, j++)
  double pt = sqrt(px[j]*px[j]+py[j]*py[j])
```

Visualization : canvas

```
TCanvas*myC1 = new TCanvas("myC1","A
Canvas",10,10,800,600);
// 定义一个画布
myC1->Divide(2,1); // 将画布分成两部分
myC1->cd(1); // 进入第一部分
f1->Draw();
myC1->cd(2); // 进入第二部分
f2->Draw();
// f1 f2 可以是直方图 也可以是 TF(function in ROOT)
myC1->SaveAs("myex33.png"); // 保存画出的图为像素图
myC1->SaveAs("myex33.eps"); // 保存画出的图为矢量图
myC1->SaveAs("myex33.pdf"); // 保存画出的图为 PDF
myC1->SaveAs(“myex33.root”); // 保存画出的图为 root binary 文件
```

ROOT I/O

- ROOT binary dataformat
- Interactive mode
  - root -l unweighted_events.root
  - .ls : list all object in root file
- create : TFile* file = new TFile("weighted_events.root", “RECREATE”)
- read : TFile* file = new TFile("weighted_events.root", “READ”)
- write : obj->Write()
- close : file->Close()

Ways to use ROOT

- The options for root

```
bash-3.2$ root -h

usage: root [-b B] [-x X] [-e E] [-n N] [-t T] [-q Q] [-l L] [-a A] [-config CONFIG] [-h HELP] [--version VERSION] [--notebook NOTEBOOK]
            [--web WEB] [--web=<browser> WEB=<BROWSER>]
            [dir] [file:data.root] [file1.C...fileN.C]

OPTIONS:
  -b                            Run in batch mode without graphics
  -x                            Exit on exceptions
  -e                            Execute the command passed between single quotes
  -n                            Do not execute logon and logoff macros as specified in .rootrc
  -t                            Enable thread-safety and implicit multi-threading (IMT)
  -q                            Exit after processing command line macro files
  -l                            Do not show the ROOT banner
  -a                            Show the ROOT splash screen
  -config                       print ./configure options
  -h, -?, --help                Show summary of options
  --version                     Show the ROOT version
  --notebook                    Execute ROOT notebook
  --web                         Display graphics in a default web browser
  --web=<browser>               Display graphics in specified web browser
  [dir]                         if dir is a valid directory cd to it before executing
  [file:data.root]              Open the ROOT file data.root
  [file1.C...fileN.C]           Execute the the ROOT macro file1.C ... fileN.C
```

Ways to use ROOT： C++ like coding

gauss_example.C

```
include <TH1.h>
#include <TFile.h>
#include <TRandom.h>

int main() {
TH1F *histo = new TH1F (“hgaus”, “A Gauss Function”,
100, -5.0, 5.0);
TRandom rnd;
for (int i = 0; i < 10000; ++i) {
   double x = rnd.Gaus (1.5, 1.0);
   histo->Fill (x); }
TFile outfile (“gaus.root”, “RECREATE”);
histo->Write();
outfile.Close();
return 0; }
```

Ways to use ROOT： C++ like coding

gauss_example.C

Compile and run:

```
$> g++ -I `root-config --incdir` -o gausexample
gausexample.C `root-config --libs` $> ./gausexample
```

```
bash-3.2$ root-config --incdir
/Users/shiki/ROOT/root_build/include
bash-3.2$ root-config --libs
-L/Users/shiki/ROOT/root_build/lib -lCore -lImt -lRIO -lNet -lHist -lGraf -lGraf3d -lGpad -lROOTVecOps
 -lTree -lTreePlayer -lRint -lPostscript -lMatrix -lPhysics -lMathCore -lThread -lMultiProc -lROOTData
Frame -Wl,-rpath,/Users/shiki/ROOT/root_build/lib -stdlib=libc++ -lpthread -lm -ldl
```

Ways to use ROOT： Macro / script

- Unnamed script
  example1.C

```
{cout << "This is the first example!" << endl;
int Num=5;
for (int i=0;i<Num;i++) {
cout<< "i=" << i << endl;
}
}
```

- Run unnamed script

```
root example1.C
```

Ways to use ROOT： Macro / script

- Named script : example2.C

类似 C++ 程序会执行” main()” 函数，
对于一个名为” xxx.C” 的 ROOT 脚本文件，如果在内部定义函数，
那么 ROOT 执行的目标则会是” xxx()” 函数

```
void func1_in_example2(int i){
 cout<< "i=" << i << endl;
}
void example2(){
cout << "This is the second example!" << endl; int
Num=10;
for (int i=0;i<Num;i++) {
    func1_in_example2(i)
}
```

- Run named script: `root example2.C`

Ways to use ROOT： Macro / script

- Named script : example2.C

类似 C++ 程序会执行” main()” 函数，
对于一个名为” xxx.C” 的 ROOT 脚本文件，如果在内部定义函数，
那么 ROOT 执行的目标则会是” xxx()” 函数

```
void func1_in_example2(int i){
 cout<< "i=" << i << endl;
}
void example2(){
cout << "This is the second example!" << endl; int
Num=10;
for (int i=0;i<Num;i++) {
    func1_in_example2(i)
}
```

- Run named script: `root example2.C`

ROOT interactive mode

- root

```
root [0] TFile *file0 = TFile::Open("gaus.root")
root [1] hgaus.Draw()
root [2] hgaus.Draw(“E”)
```

- run macro / script / c++ like code in interactive mode
  - unnamed macro : .x example1.C
  - named macro : .L example2.C
    example2()
  - c++ like : .L gauss_example.C
    main()

ROOT “interactive mode running in a non-interactive way”

- Example : TTree MakeClass MyClass
- gSystem->CompileMacro(" MyClass.C","g0ck")

  (k : keep shared library even when session is over)

  gSystem.Load(MyClass_C’)

```
MyClass* m = new MyClass()
```

ROOT in Python

- https://root.cern/manual/python/

Getting started

When ROOT is installed, you can use PyROOT both from the Python prompt and from a Python script. The entry point to PyROOT is the `ROOT` module, which you must import first:

```
import ROOT
```

Then you can access the ROOT C++ classes, functions, etc. via the `ROOT` module.

Example

This example shows how you can create a histogram (an instance of the `TH1F` class) and randomly fill it with a gaussian distribution.

```
h = ROOT.TH1F("myHist", "myTitle", 64, -4, 4)
h.FillRandom("gaus")
```

Demonstration

https://root.cern.ch/doc/master/hvector_8C.html

RootExAnalysis

https://cp3.irmp.ucl.ac.be/projects/ExRootAnalysis/wiki/UserManual

```
{
  // Load shared library
  gSystem->Load("lib/libExRootAnalysis.so");
  gSystem->Load("libPhysics");

  // Create chain of root trees
  TChain chain("LHCO");
  chain.Add("pgs_events.root");

  // Create object of class ExRootTreeReader
  ExRootTreeReader *treeReader = new ExRootTreeReader(&chain);
  Long64_t numberOfEntries = treeReader->GetEntries();

  // Get pointers to branches used in this analysis
  TClonesArray *branchJet = treeReader->UseBranch("Jet");
  TClonesArray *branchElectron = treeReader->UseBranch("Electron");

  // Book histograms
  TH1 *histJetPT = new TH1F("jet_pt", "jet P_{T}", 100, 0.0, 100.0);
  TH1 *histMass = new TH1F("mass", "M_{inv}(e_{1}, e_{2})", 100, 40.0, 140.0);

  // Loop over all events
  for(Int_t entry = 0; entry < numberOfEntries; ++entry) {

    // Load selected branches with data from specified event
    treeReader->ReadEntry(entry);

    // If event contains at least 1 jet
    if(branchJet->GetEntries() > 0) {

      // Take first jet
      TRootJet *jet = (TRootJet*) branchJet->At(0);

      // Plot jet transverse momentum
      histJetPT->Fill(jet->PT);
```

Delphes

https://cp3.irmp.ucl.ac.be/projects/delphes/wiki/WorkBook

https://cp3.irmp.ucl.ac.be/projects/delphes/wiki/WorkBook/RootTreeDescription

```
void Example1(const char *inputFile)
{
  gSystem->Load("libDelphes");

  // Create chain of root trees
  TChain chain("Delphes");
  chain.Add(inputFile);

  // Create object of class ExRootTreeReader
  ExRootTreeReader *treeReader = new ExRootTreeReader(&chain);
  Long64_t numberOfEntries = treeReader->GetEntries();

  // Get pointers to branches used in this analysis
  TClonesArray *branchJet = treeReader->UseBranch("Jet");
  TClonesArray *branchElectron = treeReader->UseBranch("Electron");

  // Book histograms
  TH1 *histJetPT = new TH1F("jet_pt", "jet P_{T}", 100, 0.0, 100.0);
  TH1 *histMass = new TH1F("mass", "M_{inv}(e_{1}, e_{2})", 100, 40.0, 140.0);
```

More examples in macro or python script at :
Delphes/examples

Homework

- Generate leading order process each separately
  1. p p > e+ e- (all contributions)
  2. p p > z, z > e+ e- (z is on-shell)
  3. p p > e+ e- $z (forbids s-channel z to be on-shell)

通过 m(e+e-) 的柱状图，验证过程 1= 过程 2+ 过程 3
柱状图要求归一化到散射截面

1. m(e+e-) 通过 LHE event 得到 (ExRootanalysis)
2. m(e+e-) 通过 Delphes 中的 Electron (的四动量) 得到