ROOT Basics

李刚

ligang@ihep.ac.cn

IHEP School of Computing 2020

2020 Aug 24-25

说明

A. ROOT 的功能很多、很强大——术

B. 这里只是从数据分析用户角度出发介绍部分 ROOT 的功能，ROOT 的一个子集

C. 至于其它很多功能，只提到一点

D. 如果其它需求的，需要自学。

E. 学点统计——道

官网：https://ROOT.cern.ch/

最有用的信息：https://ROOT.cern/ROOT/html534/ClassIndex.html

论坛：https://ROOT-forum.cern.ch/

例子：/cefs/higgs/lig/Tutorial/root

Some basic linux commands

Linux command	What it does...
`ls`	List contents of a directory
`pwd`	Show present working directory
`mkdir test`	Make a new directory called test
`cd test`	Change to directory test
`cp file1.txt file2.txt`	Copy file1.txt to file2.txt
`mv file1.txt file3.txt`	Move file1.txt to file3.txt

`cat file2.txt`
`less file2.txt`
`more file2.txt`
	Print the contents of a file to the screen

`emacs -nw`
`vi`
`pico`
`...`
	Console text editors (no extra window pops up)

`emacs`
`xemacs`
`nedit`
`gedit`
`...`
	GUI text editors (extra window pops up)

提纲

1. 数据分析做什么？用什么工具?
2. ROOT能做什么?
3. ROOT的使用模式：
    1. 命令行交互
    2. Macros
    3. 用户编译代码
4. ROOT中的对象：histogram, tree, functions
5. ROOT拟合 — 统计估计、假设检验
6. ROOT画图
7. ROOT其它功能
8. 小结和继续学习

ROOT 是工具的森林，大部分用户只学了其中几棵树
对ROOT理解的越深刻，效率越高，在数据分析中拥有更大的自由

数据分析做什么？用什么工具？ROOT

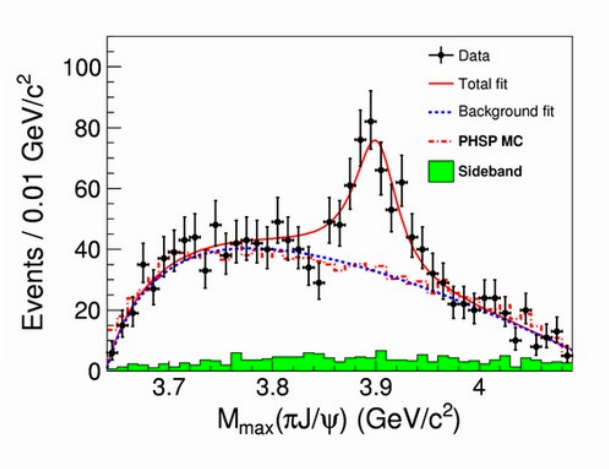
事例选择：寻找和使用 cuts

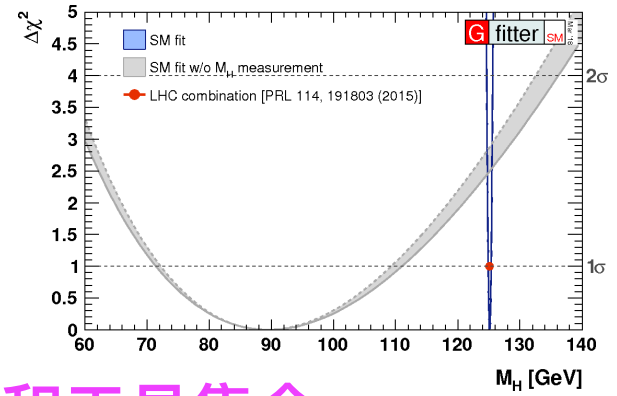
统计估计

- ➢ 发现数据中的新pattern
- ➢ 新粒子、新现象
- ➢ 给出定量的估计
- ➢ 设限、置信区间，…

假设检验

- ➢ 用数据说话，排除或者验证理论





很多工具都能满足你的需求
背后都是一样的统计原理
ROOT最自然、高效的选择

数据分析框架和工具集合

ROOT，MatLab, Python, R, …

什么是数据分析

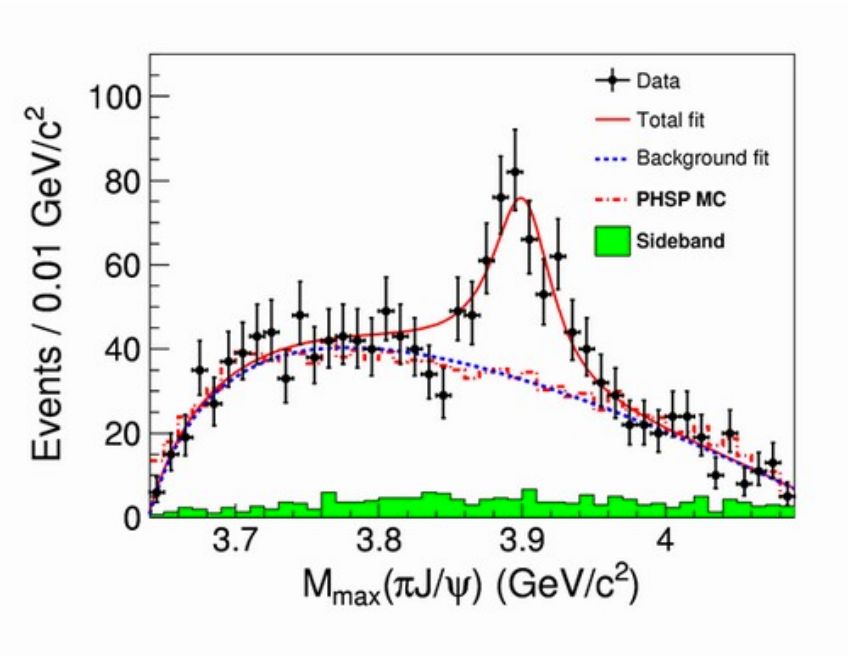
你的数据

```
3 *    150.142 *    1.45592 *    4.66344 *    146.706 *    1.02172 *
4 *    149.942 *    -10.342 *    11.0689 *    148.325 *   0.854095 *
5 *    150.185 *    17.0842 *   -12.1425 *    143.101 *   0.902942 *
6 *    150.018 *    5.19219 *    7.78532 *    148.594 *    1.06437 *
7 *    150.052 *    7.54787 *   -7.43332 *    144.446 *   0.972789 *
8 *    150.071 *   0.231547 *  -0.021123 *    147.784 *   0.928199 *
```

经过一个复杂观察、设计和优化过程

Turning your data into something that is human understandable and therefore meaningful

你的结果



BESIII Zc(3900)

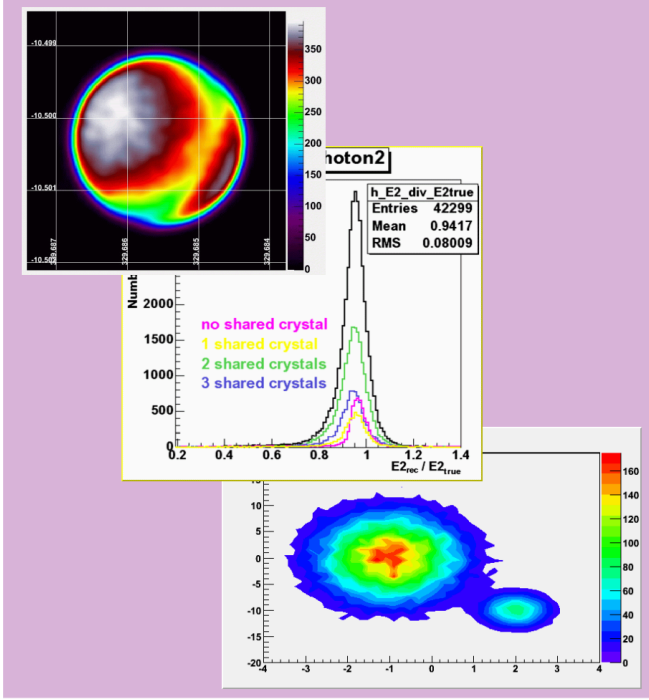
ROOT能做什么?

- Save data （O）
- Access data （I）
- Mine data （寻找新东西）
- Publish results （画图）
- Run interactively or build your own application （各种模式的高效结合）
- Use ROOT within other languages

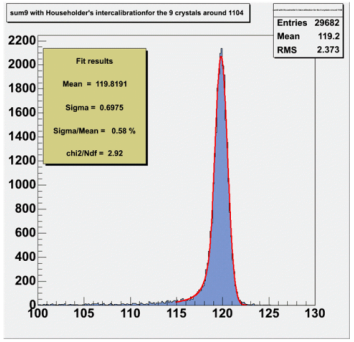
ROOT能做什么(部分)?

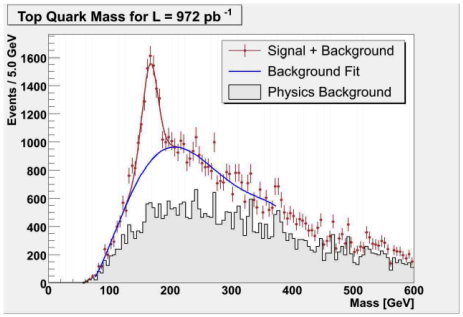
- (some of) What ROOT can do:

Plotting

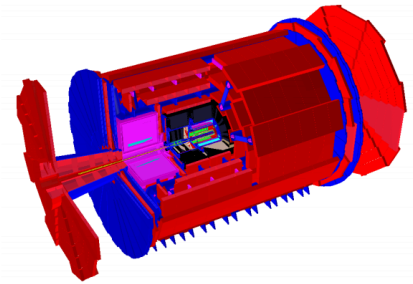


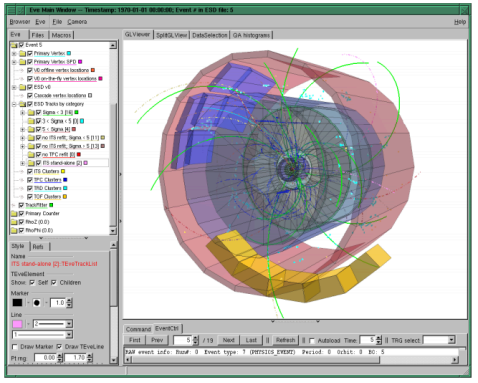
Fitting / Computation





Detector Visualization





ROOT使用模式

- Command line interactive
- Scripts: w/i name and w/o name
- MakeClass()
- Compile user code

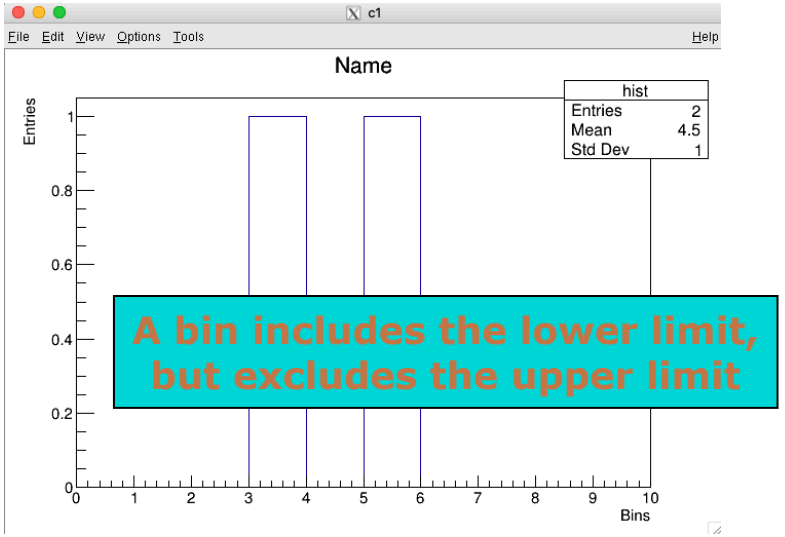
ROOT中的对象

- **Data**
  - **Histogram**
  - **Tree/Ntuples**
  - **Graph(errors)**
- **Functions**
- **Canvas, Pad, File, Random numbers, Fitter, Minuit, …**

Data-Histograms (1)

- Binned data – 可能是对物理学家来说最重要的数据形式
- 一般和概率密度函数可以直接联系起来
- Examples： TH1F （1 维，浮点型）
  - root [ ] TH1F *h = new TH1F("hist", “Name;Bins;Entries”, 10, 0, 10);
    - “hist” is a (unique) name
    - “Name;Bins;Entries” are title and the x and y labels
    - 10: is number of bins
    - 0, 10, limits on the x axis. First bin is from 0, 1, …
  - Fill the histogram
    - root [ ] h->Fill(3.5);
    - root [ ] h->Fill(5.5);
  - Draw the Histogram

    root [ ] h->Draw();



Data-Histograms(2)

```
root [ ] TH1F h("h","h",80,-40,40) ;
root [ ] TRandom r;
root [ ] for (i=0;i<10000;i++) { h.Fill(r.Gaus(0,7));}
root [ ] h.Draw()
```

- Rebinning

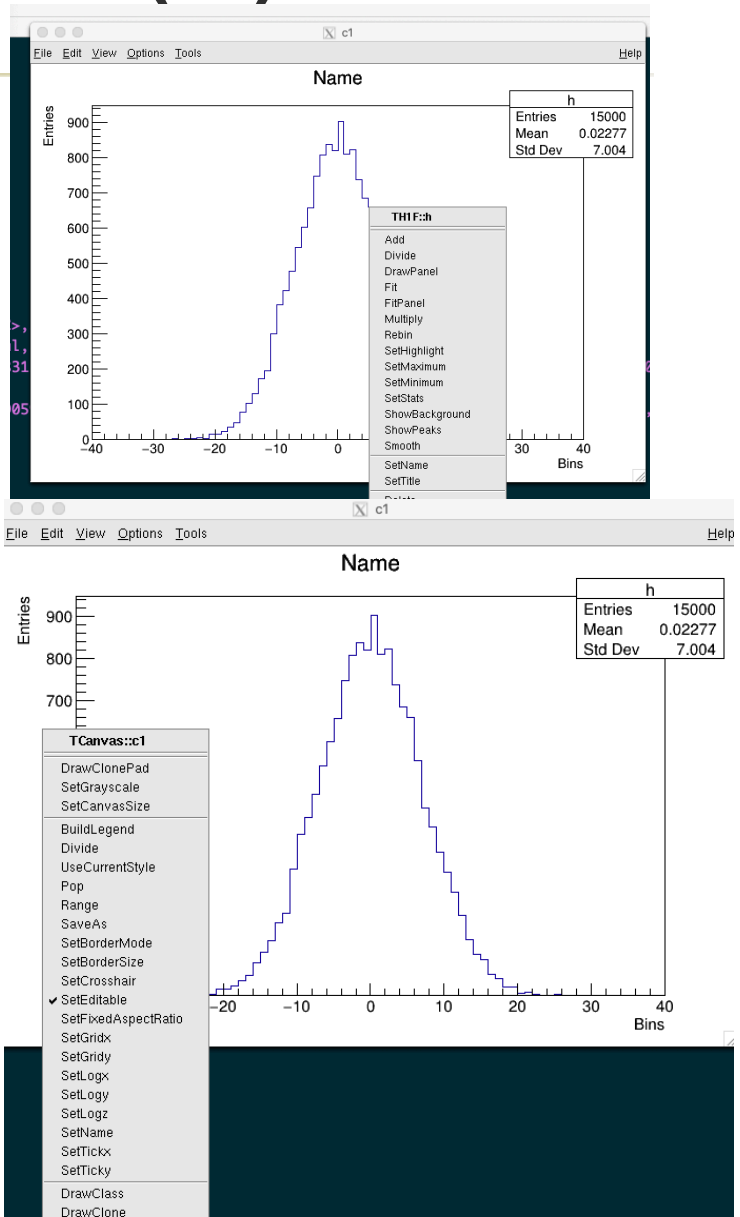
  root [ ] h.Rebin(2)

- Change ranges/canvas
  - with the mouse, very easy!
  - with the context menu
  - command line

  root [ ] h.GetXaxis()-> SetRangeUser(2, 5)

- Log-view
  - right-click in the white area at the side of the canvas and select SetLogx (SetLogy)
  - command line

  root [ ] gPad->SetLogy()

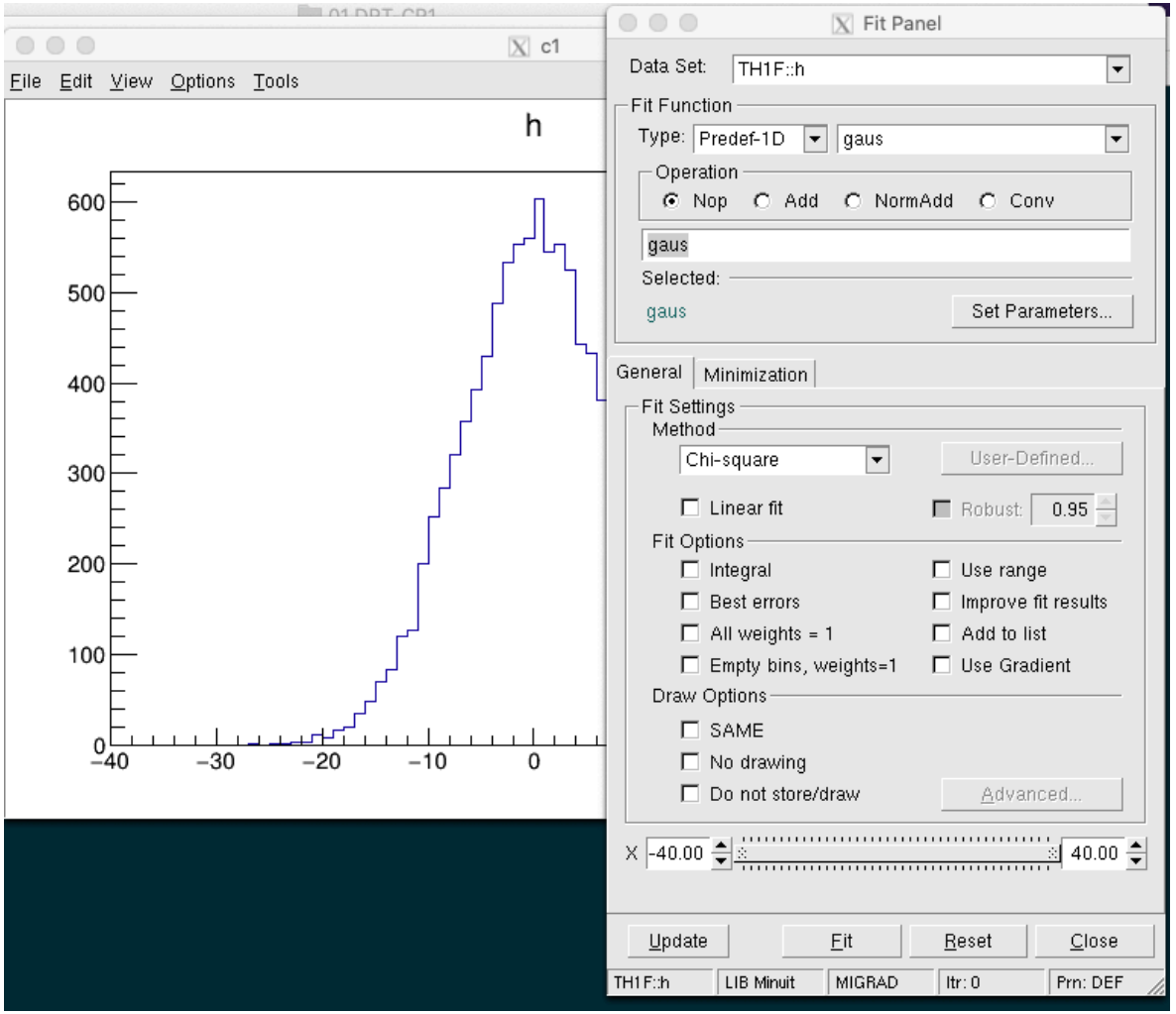


Data-Histogram fit(3)

- Interactive
  - – Right click on the histogram and choose "fit panel"
  - – Select function and click fit
  - – Fit parameters
    - are printed in command line
    - in the canvas: options - fit parameters
- Command line

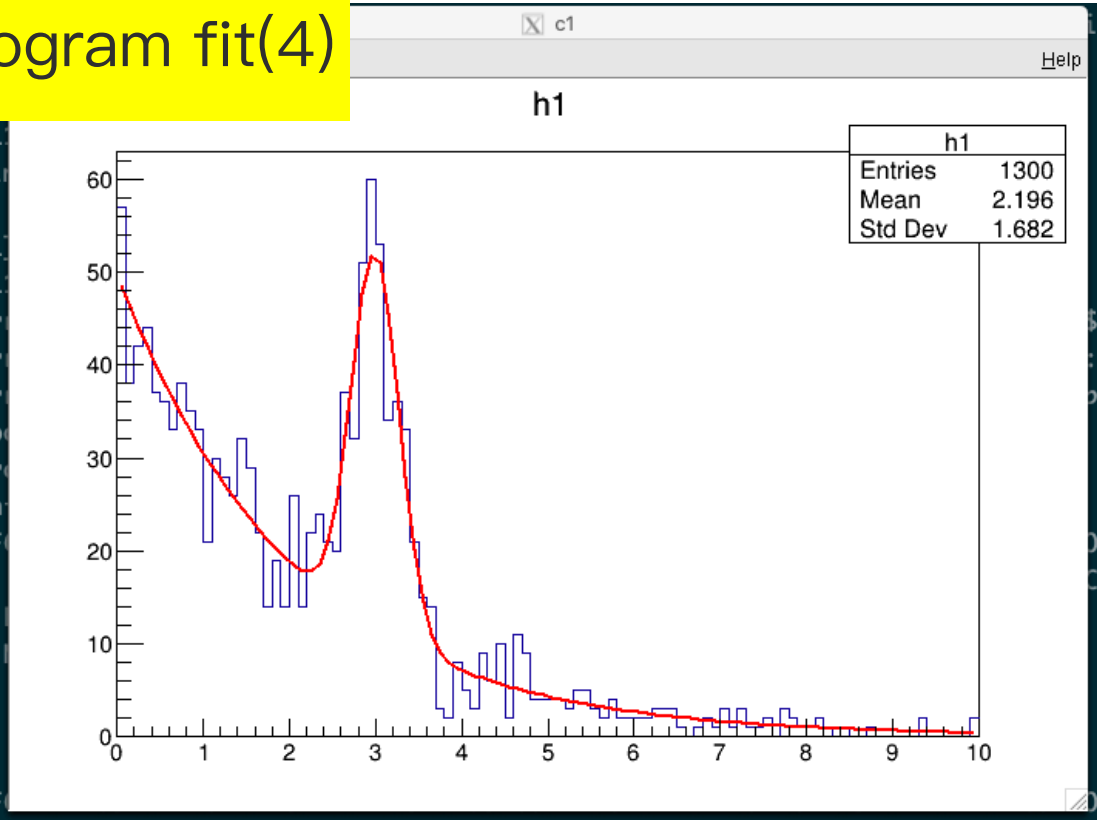
root [ ] h->Fit("gaus")
– Other predefined functions polN (N = 0..9), expo, landau

- Try to fit the histogram with different functions.



Data–Histogram fit(4)

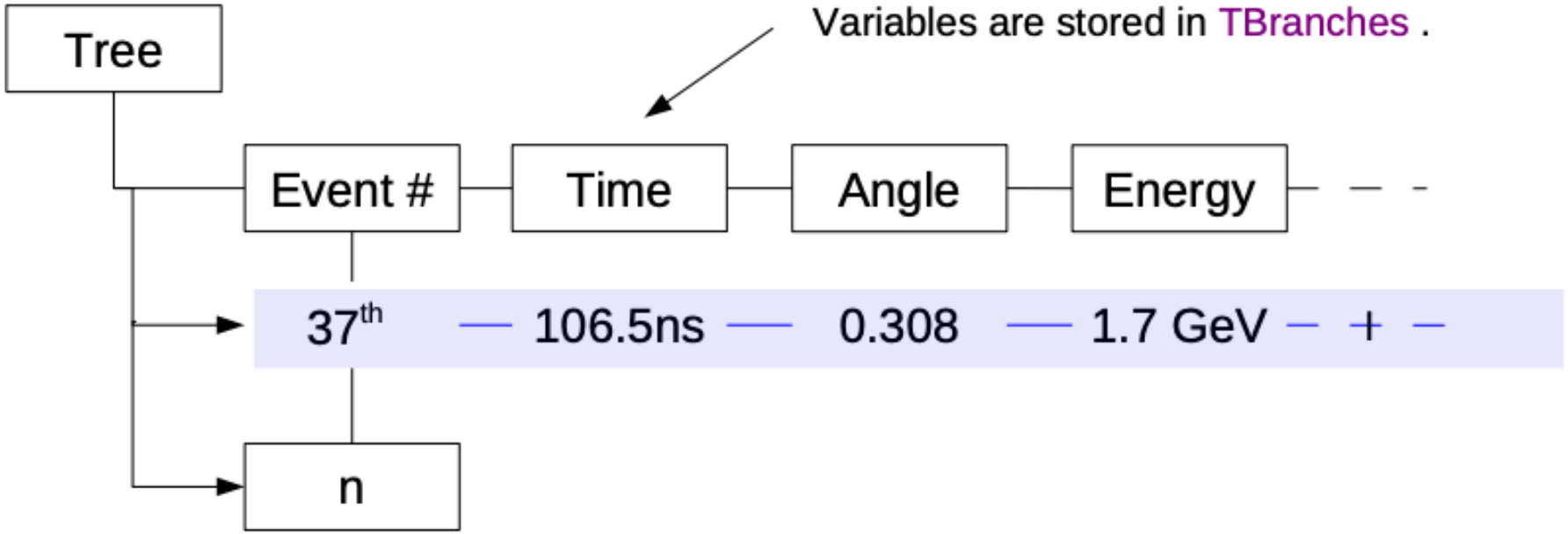
Using f.C




```
                  EDM=9.21995e-10    STRATEGY= 1      ERROR MATRIX ACCURATE
  EXT PARAMETER                                          STEP         FIRST
  NO.   NAME      VALUE            ERROR                 SIZE      DERIVATIVE
   1  Constant     3.92357e+00   4.50038e-02   2.51752e-04   1.16715e-03
   2  Slope       -4.84374e-01   1.69081e-02   9.45620e-05   3.57764e-03
 FCN=50.0767 FROM MIGRAD    STATUS=CONVERGED     117 CALLS         118 TOTAL
                  EDM=3.80564e-08    STRATEGY= 1  ERROR MATRIX UNCERTAINTY   1.8 per cent
  EXT PARAMETER                                          STEP         FIRST
  NO.   NAME      VALUE            ERROR                 SIZE      DERIVATIVE
   1  p0           4.04778e+01   3.61764e+00  -2.07071e-03   1.13822e-05
   2  p1           2.99642e+00   2.49402e-02   2.86369e-05   7.23451e-03
   3  p2           2.79666e-01   2.20363e-02   1.67157e-05   5.37487e-03
   4  p3           3.90804e+00   4.67380e-02   9.40009e-05  -4.76800e-03
   5  p4          -4.88215e-01   1.75470e-02  -1.91494e-05  -9.11828e-03
                              ERR DEF= 0.5
number of signal events = 283.757 +/- 22.7457
```

Data - TTree (1)

- TTree 是最ROOT 中重要的一种数据形式，和表格类似；可以同时操作多个数据变量—行操作、列操作
- 可以在交互式和批处理模式根据不同的选择条件进行绘图和计算
- 内部压缩算法，对便于数据存储和IO



Data - TTree (2)

Useful commands

tree->Show(0);
tree->Print();
tree->Scan(“”);
tree->Draw(“x:y”)

beside quick ones

new TBrowser

TBrowser a

```
1  #include "TRandom.h"
2  #include "TFile.h"
3  #include "TTree.h"
4
5  void SimpleTree(const char * filename= "tree.root") {
6
7     TTree data("tree","Example TTree");
8     double x, y, z, t;
9     data.Branch("x",&x,"x/D");
10    data.Branch("y",&y,"y/D");
11    data.Branch("z",&z,"z/D");
12    data.Branch("t",&t,"t/D");
13
14 // fill it with random data
15    for (int i = 0; i<10000; ++i) {
16       x = gRandom->Uniform(-10,10);
17       y = gRandom->Gaus(0,5);
18       z = gRandom->Exp(10);
19       t = gRandom->Landau(0,2);
20
21       data.Fill();
22    }
23 // write in a file
24    TFile f(filename,"RECREATE");
25    data.Write();
26    f.Close();
27
28 }
```

Define

Fill

Save

Data - TTree (3)

Efficient and powerful commands with a tree

tree->MakeClass(“tree”);

tree->MakeSelector(“testSelector”);

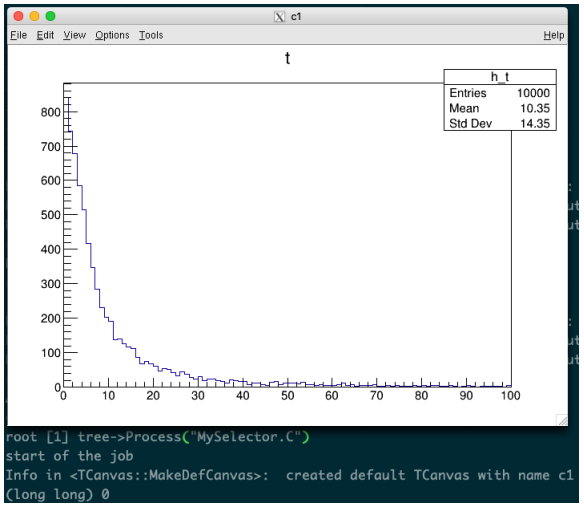
自动产生代码：root[] tree->MakeSelector("MySelector.C");

稍作修改：

在 MySelector::SlaveBegin 定义histograms

在 MySelector::process() 填写histograms

在 MySelector::Terminate 画 histograms.



Functions

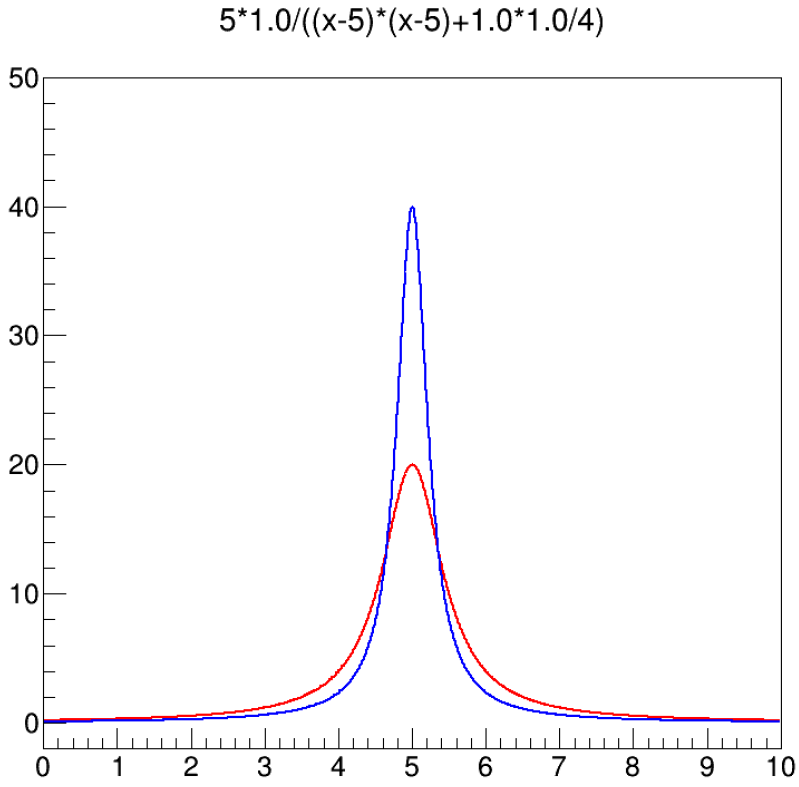
- Function 是 ROOT 提供的最基本功能之一
- 研究的最开始往往即使文献中的一个公式（函数），尤其复杂函数和非解析函数
- 用两种方式来实现 Breit-Wigner 公式
- $$BW = \frac{M\Gamma}{(m - m_0)^2 + \frac{\Gamma^2}{4}}$$
- 有无参数
- 求值
- 求导
- 求积分

```
TF1 * f1 = new TF1("f1","5*1.0/((x-5)*(x-5)+1.0*1.0/4)",0,10);
f1->SetNpx(1000);
f1->Draw();

TF1 * fp = new TF1("fp","[0]*[1]/((x-[0])*(x-[0])+[1]*[1]/4)",0,10);
fp->SetNpx(1000);
fp->SetParameters(5,0.5);
fp->Draw("same");
fp->SetLineColor(kBlue);
```

Functions

```
std::cout << "Value of f(x) at x = 5      is  " << fp->Eval(5.) << std::endl;
std::cout << "Derivative of f(x) at x = 4 is  " << fp->Derivative(4.) << std::endl;
std::cout << "Derivative of f(x) at x = 6 is  " << fp->Derivative(6.) << std::endl;
std::cout << "Integral of f(x) in [4, 5]  is  " << fp->Integral(4,6)  << std::endl;
std::cout << "Integral of f(x) in [0,10]  is  " << fp->Integral(0,10)  << std::endl;
```

```
root [0]
Processing plotBW.C...
Info in <TCanvas::MakeDefCanvas>:  created def
Value of f(x) at x = 5      is  40
Derivative of f(x) at x = 4 is  4.42907
Derivative of f(x) at x = 6 is  -4.42907
Integral of f(x) in [4, 5]  is  26.5164
Integral of f(x) in [0,10]  is  30.4168
```



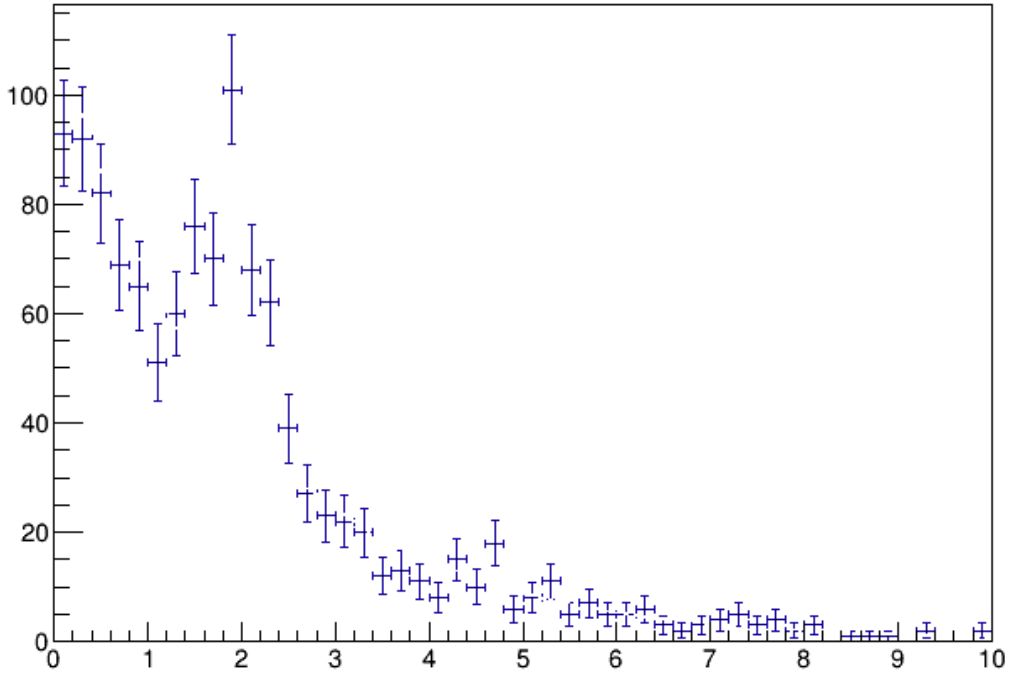
数据分析的核心——
统计估计、假设检验（拟合）

- 当我们说拟合的时候，我们在说什么?
- 大学数学到底是数学分析重要，还是统计、概率更重要?
- Data mining
  - 统计估计(statistical estimation)是统计推断的一种形式，统计估计的方法是用样本的函数来估计总体的分布函数、分布参数或数字特征。
    - 例子：从样本分布中估计某个粒子的质量、宽度、事例数，某个特征，比如不对称等
  - 假设检验(hypothesis testing)，又称统计假设检验：需要对结果进行假设，然后拿样本数据去验证这个假设。
    - 显著性
    - 排除某个理论模型
  - 在数据中寻找新 Pattern
- 更多的希望大家学习 RooFit & RooStats 的 tutorial
- 下来我们看例子

一个典型的 ROOT fit

从一个直方图里确定 peak 和平滑本底里包含的事例数及误差（200+1000）

```
TH1D * h1 = new TH1D("h1","h1",50,0,10);

for (int i = 0; i < 200; ++i) {
   h1->Fill(gRandom->Gaus(2,0.3));
}
for (int i = 0; i < 1000; ++i) {
   h1->Fill(gRandom->Exp(2));
}
```



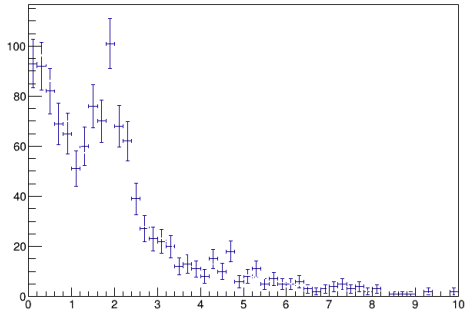
为了拟合成功，采用尝试拟合的方法得到初值

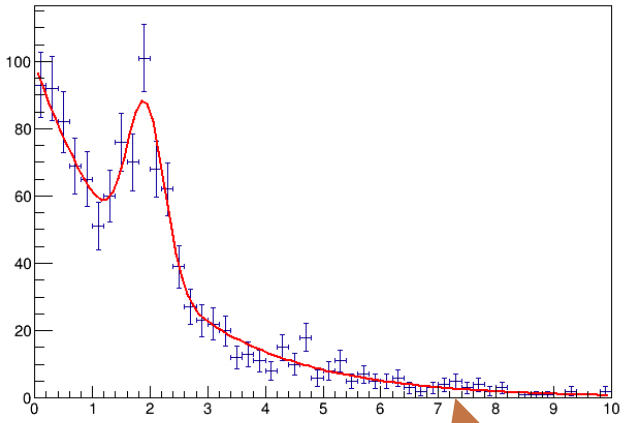
```
// fit first a single gaussian in range [1,3]
TFitResultPtr r1 = h1->Fit("gaus","S","",1,3);  // first fit of gaussian
TFitResultPtr r2 = h1->Fit("expo","S");  // first exponential

TF1 * f1 = new TF1("fitFunc","gaus(0)+expo(3)");

// get parameters and set in global TF1
// parameters of first gaussian
f1->SetParameter(0,r1->Parameter(0));
f1->SetParameter(1,r1->Parameter(1));
f1->SetParameter(2,r1->Parameter(2));
// parameters of expotentail
f1->SetParameter(3,r2->Parameter(0));
f1->SetParameter(4,r2->Parameter(1));
```

信号

本底

ROOT 自己的拟合因为没有太多的封装，
可以控制大部分细节，下面还会看到





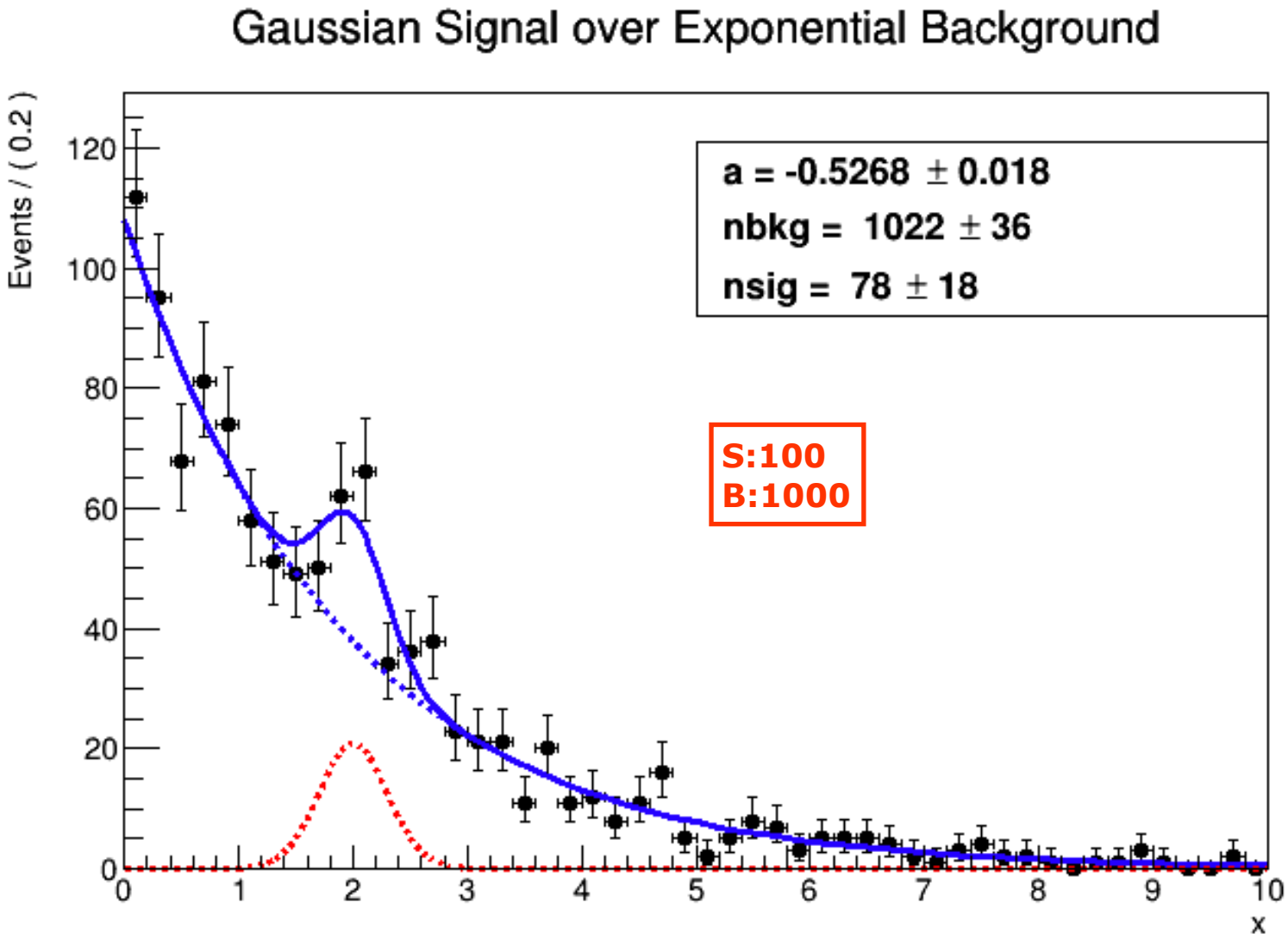
```
FCN=20.4791 FROM MIGRAD    STATUS=CONVERGED     153 CALLS         154 TOTAL
                     EDM=3.63954e-09    STRATEGY= 1      ERROR MATRIX ACCURATE
EXT PARAMETER                                          STEP         FIRST
NO.   NAME      VALUE            ERROR          SIZE      DERIVATIVE
 1  p0           4.98546e+01   6.40352e+00   1.64489e-02   5.20252e-06
 2  p1           1.92384e+00   4.36789e-02   1.35136e-04  -5.31489e-04
 3  p2           3.22704e-01   3.97793e-02   9.71043e-05   9.72725e-04
 4  p3           4.59526e+00   5.21273e-02   1.06341e-04   1.19264e-03
 5  p4          -4.94114e-01   1.74355e-02   3.85553e-05   1.63530e-04
                               ERR DEF= 0.5
```

```
// fitting
h1->Draw("E1");
TFitResultPtr res = h1->Fit(f1,"SL");


// compute number of signal events
TMatrixDSym cov = res->GetCovarianceMatrix();
TMatrixDSym covPeak = cov.GetSub(0,2,0,2);

// fitted gaussian function
TF1 * peakFunc = new TF1("peakFunc","gaus");
peakFunc->SetParameter(0, f1->GetParameter(0));
peakFunc->SetParameter(1, f1->GetParameter(1));
peakFunc->SetParameter(2, f1->GetParameter(2));


double nsignal = peakFunc->Integral(0,10) / h1->GetBinWidth(1);
double err     = peakFunc->IntegralError(0,10,peakFunc->GetParameters(), covPeak.GetMatrixArray()) / h1->GetBinWidth(1);
```

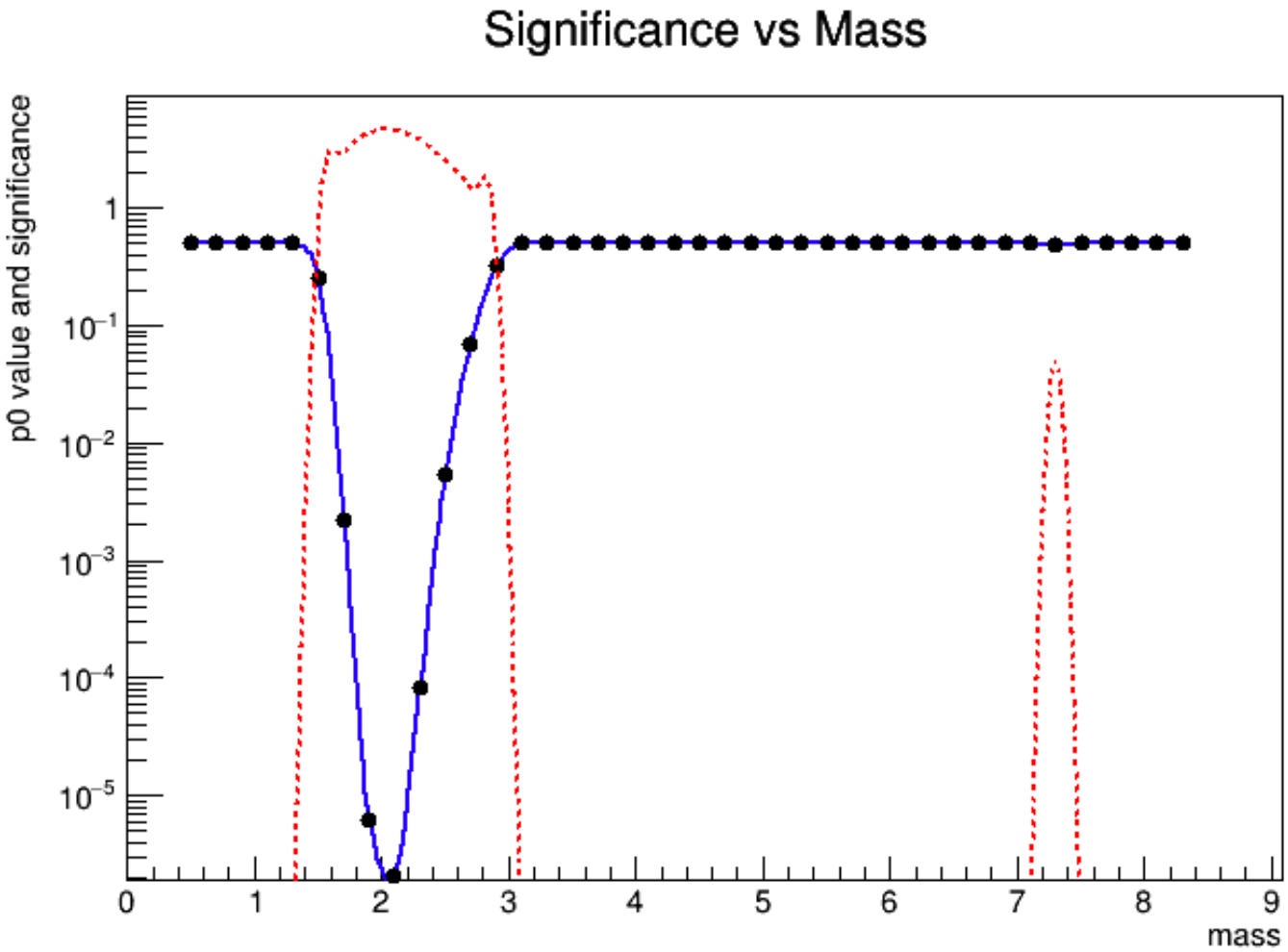
通过调用拟合参数的结果和误差矩阵，计算事例数（bin width）

number of signal events = 201.636 +/- 26.2222

一个 RooFit 拟合、计算 significance 的



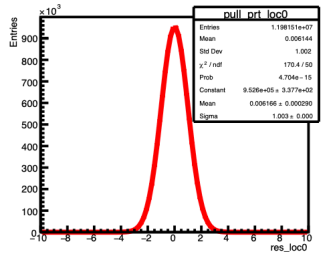
P value/significance vs mass

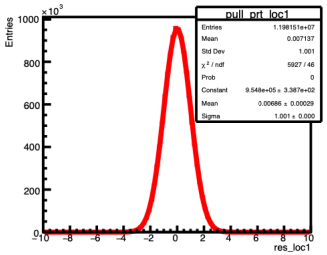


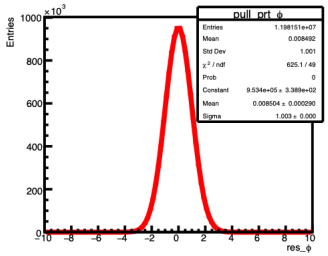
ROOT画图

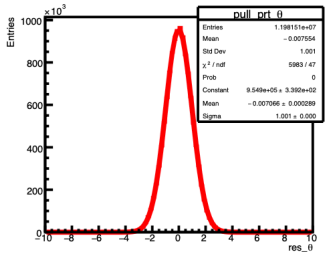
- 画图非常花时间，画漂亮的图尤其花时间
- 关键在于两点
  - 花较少时间画出审美还行的的图，用于内部报告——重用成熟的画图脚本 （正式的合作组一般都要子集的**PlotStyle**）
  - 把重要结果画出真与美完美结合的效果——个人发挥
    - 元素突出
    - 关键信息完备：**legend, axis name/title, ...**
    - 结构合理

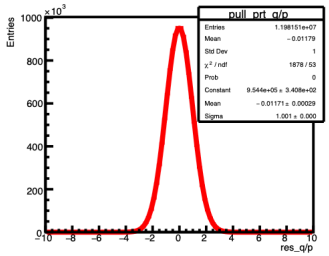
用于组会讨论的图（附简表）

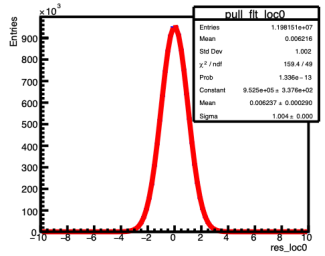


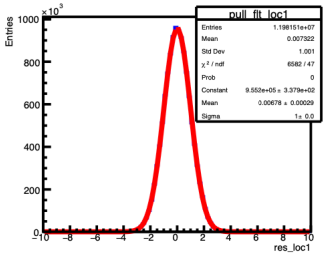


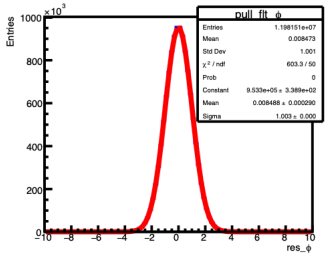


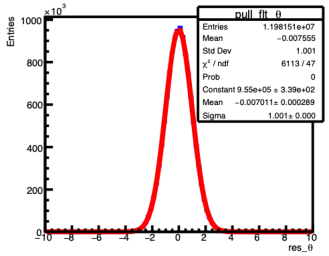


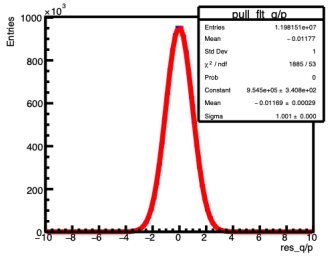


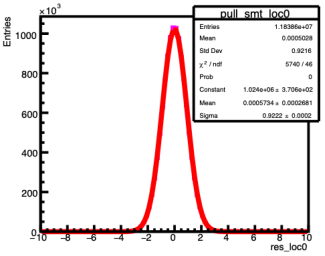


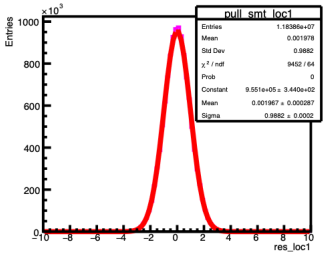


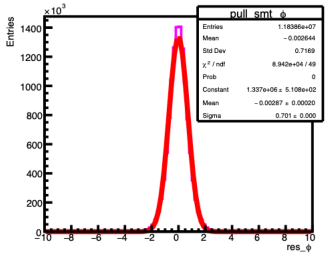


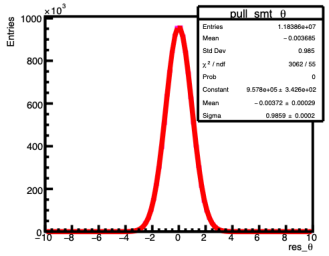


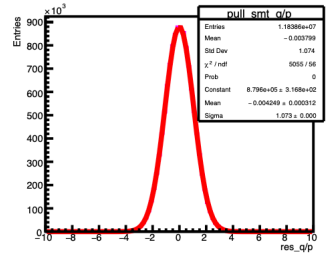




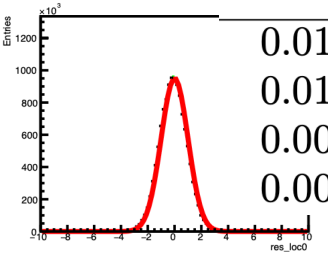


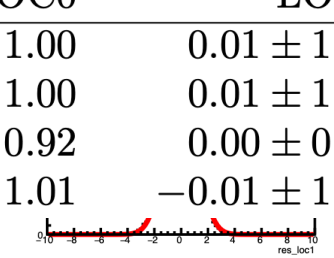


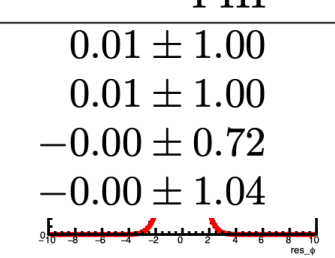


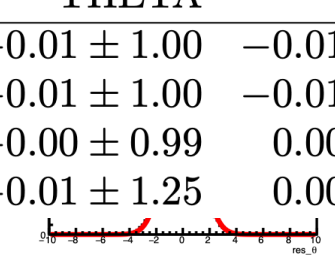


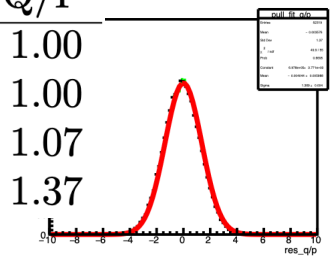
	LOC0	LOC1	PHI	THETA	Q/P
	0.01 ± 1.00	0.01 ± 1.00	0.01 ± 1.00	-0.01 ± 1.00	-0.01 ± 1.00
	0.01 ± 1.00	0.01 ± 1.00	0.01 ± 1.00	-0.01 ± 1.00	-0.01 ± 1.00
	0.00 ± 0.92	0.00 ± 0.99	-0.00 ± 0.72	-0.00 ± 0.99	0.00 ± 1.07
	0.00 ± 1.01	-0.01 ± 1.14	-0.00 ± 1.04	-0.01 ± 1.25	0.00 ± 1.37





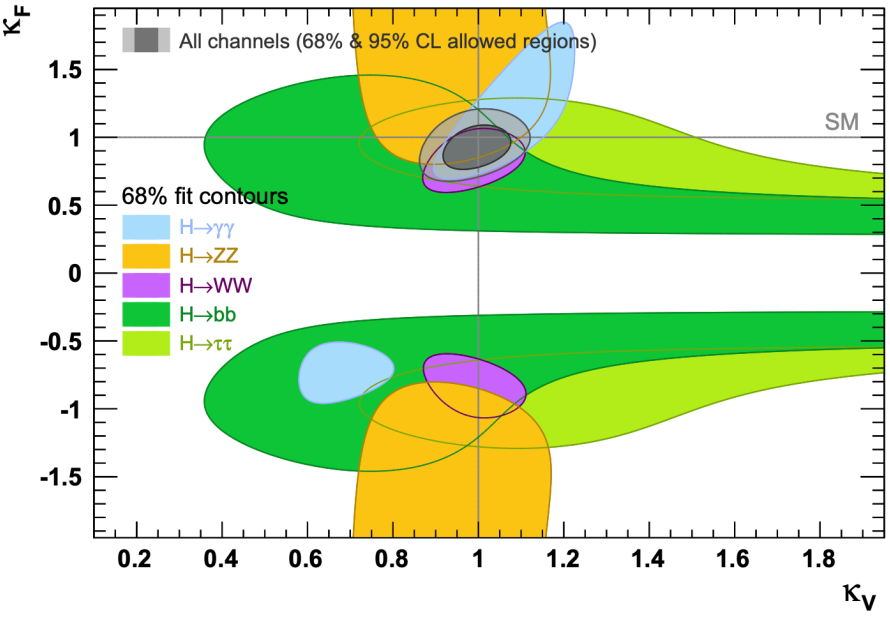


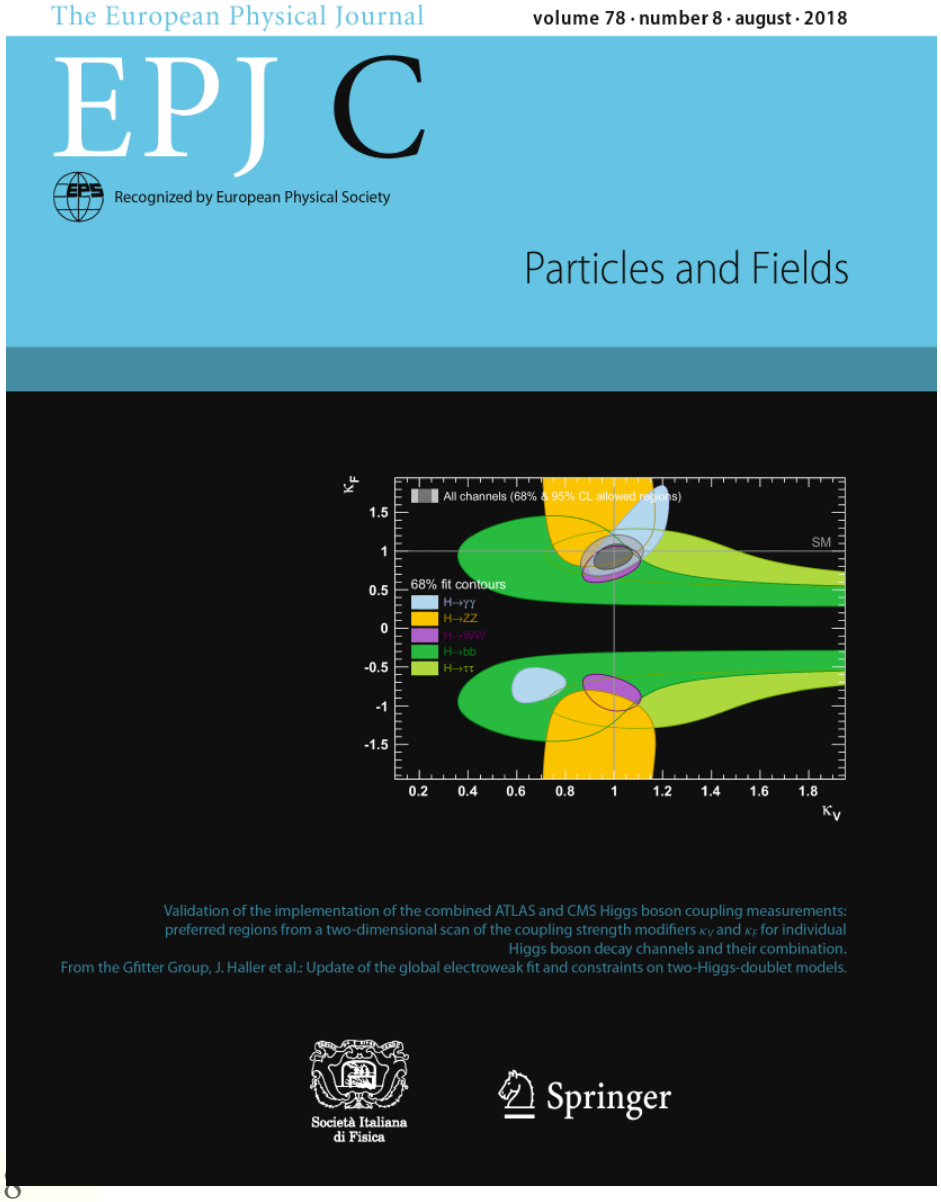




发表文章的时候，要做好准备，你的图会上杂志封面

要让数年的辛苦工作，
以更具美感的形式呈现出来





提示

- **ROOT** 命令行非常有用，能快速的让你看到数据中的一些特征，如果写 **script** 的话，耗时较长，导致无法很多本来能看到的有趣特征。
- **ROOT** 命令行解释**C++**，但是语法要求比较松。不要带坏了你的编程风格。
  - 行尾不需要“；”
  - 变量类型过于灵活
  - “.” 和 “->” 不区分
- **script** 也有类似的问题

小结

- **ROOT** 功能强大丰富，一个人不会用到全部功能。建议如下：
  - 浏览大致都有哪些功能以备用
  - 需要熟练掌握、深刻理解的：比如 **Histogram, Tree, Graphs, Function, RooFit&RooStats, ...**
  - 特殊的工具比如：**TGeo**， **TEve**， **TMVA**，等根据工作需要，**ROOT** 恐怕也无法满足你的需求，需要拓展式学习
  - **TGeo**：探测器几何构建 —— **DD4hep**
  - **TEve**： 数据和探测器可视化
  - **TMVA**： 机器学习的各种框架
- 基本的功能：直方图，函数，**Tree**， **Graph**，拟合、绘图 必须数量掌握
  - 尤其是 **Tree** 的 **MakeClass(), MakeSelector()**等功能，会大大提高工作效率
- 时刻记着数据分析的目标：发现新数据中的模式、统计估计、假设检验